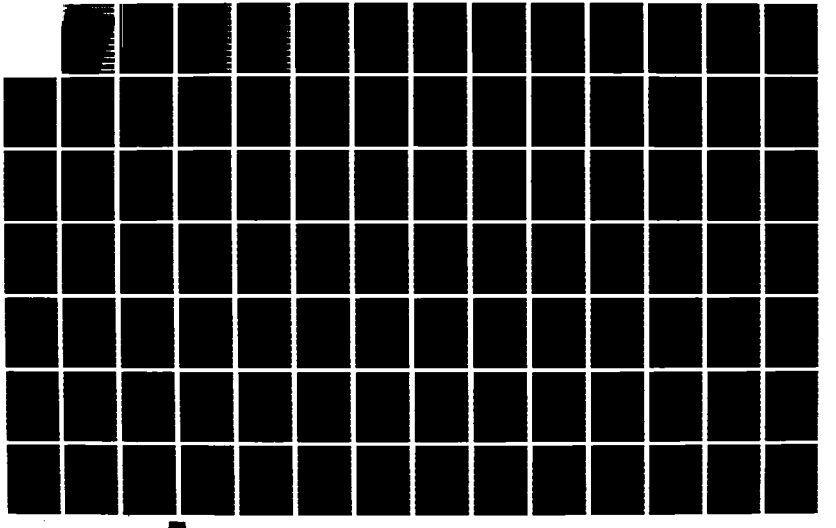
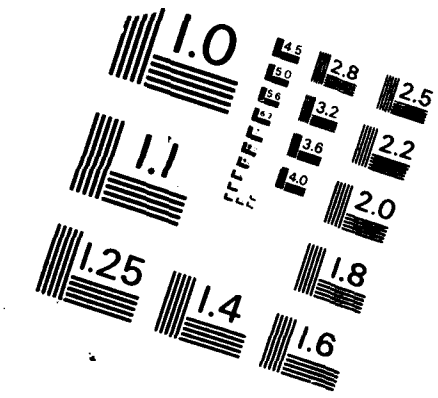
AD-A170 871 PLAN RECOGNITION AND DISCOURSE ANALYSIS: AN INTEGRATED APPROACH FOR UNDERSTANDING DIALOGUES(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE. D J LITMAN 1985 TR-170 1/3

UNCLASSIFIED N00014-82-K-0193 F/G 5/7 NL





Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues

Diane J. Litman
Department of Computer Science
The University of Rochester
Rochester, NY 14627

TR 170
1985

Department of Computer Science
University of Rochester
Rochester, New York 14627

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: TR 170	2. GOVT ACCESSION NO.: AD-A170 871
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE *(and Subtitle)*: Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Diane J. Litman	8. CONTRACT OR GRANT NUMBER(s): N00014-82-K-0193 N00014-80-C-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department University of Rochester Rochester, NY 14627	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, A 22209	12. REPORT DATE: 1985
	13. NUMBER OF PAGES: 183
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*: Office of Naval Research Information Systems Arlington, VA 22217	15. SECURITY CLASS. *(of this report)*: unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

discourse, plan recognition, meta-plans, speech acts, coherence, subdialogues, ellipsis, equality

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

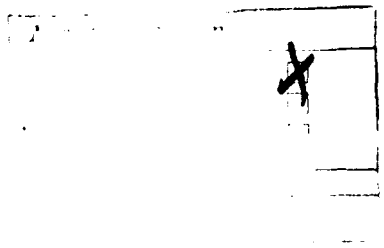
One promising computational approach to understanding dialogues has involved modeling the goals of the speakers in the domain of discourse. In general, these models work well as long as the topic follows the goal structure closely, but they have difficulty accounting for interrupting subdialogues such as clarifications and corrections. Furthermore, such models are typically unable to use many processing clues provided by the linguistic phenomena of the dialogues. (continued on next page)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (Continued)

This dissertation presents a computational theory and partial implementation of a discourse level model of dialogue understanding. The theory extends and integrates plan-based and linguistic-based approaches to language processing, arguing that such a synthesis is needed to computationally handle many discourse level phenomena present in naturally occurring dialogues. The simple, fairly syntactic results of discourse analysis (for example, explanations of phenomena in terms of very local discourse contexts as well as correlations between syntactic devices and discourse function) will be input to the plan recognition system, while the more complex inferential processes relating utterances have been totally reformulated within a plan-based framework. Such an integration has led to a new model of plan recognition, one that constructs a hierarchy of domain and meta-plans via the process of constraint satisfaction. Furthermore, the processing of the plan recognizer is explicitly coordinated with a set of linguistic clues. The resulting framework handles a wide variety of difficult linguistic phenomena (for example, interruptions, fragmental and elliptical utterances, and presence as well as absence of syntactic discourse clues), while maintaining the computational advantages of the plan-based approach. The implementation of the plan recognition aspects of this framework also addresses two difficult issues of knowledge representation inherent in any plan recognition task.



A-1

2

Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues

Diane J. Litman
Department of Computer Science
The University of Rochester
Rochester, NY 14627

TR 170
1985

This report reproduces a dissertation submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy in Computer Science at the University of Rochester, supervised by James F. Allen.

This research was supported in part by the Defense Advanced Research Projects Agency under Grant N00014-82-K-0193, the National Science Foundation under Grant DCR8351665, and the Office of Naval Research under Grant N0014-80-C-0197.

Curriculum Vitae

Diane Litman[1] was born in Manhattan on March 5, 1958. She spent her first eight years in the New York City area, then in the fall of 1966 moved with her family to the Washington suburbs of Rockville, Maryland. Luckily she was young enough to lose her New York accent.

In 1976 Ms. Litman entered the College of Mary and William in Virginia (no, it's neither single-sex nor private). On the advice of her freshman roommate she decided not to take any computer science ("Don't bother - it's just like typing classes."). Nonetheless, her physics advisor changed her mind and as a sophomore she enrolled in the introductory programming class. Soon after she dropped physics and officially declared herself a double major in the areas of Mathematics and Computer Science, dabbling in Psychology on the side.

In 1979 Ms. Litman became a member of Phi Beta Kappa, and in 1980 she received her A.B. in Mathematics and Computer Science with High Honors for senior research. (A GDI and Yankee, Ms. Litman had generally preferred studying to frat parties). During her summers Ms. Litman had helped support her schooling by working for the Interstate Commerce Commission, the National Institute of Health's Division of Computer Research and Technology, and IBM's education group.

In the fall of 1980, after the required backpacking trip through Europe, Ms. Litman began her graduate career at the Computer Science Department of the University of Rochester.

---

[1]also known as nano, the g.q., and d-squared

During those years she has been both a teaching and research assistant. In particular, she has conducted research with both the ARGOT dialogue understanding project at Rochester and the Knowledge Representation for Natural Language Understanding Group at Bolt Beranek and Newman. During 1983-1984 she was elected student representative to the faculty, a term notable for the passing of an enlightened graduate student support policy. Ms. Litman has also spent a summer at the Linguistic Society of America Summer Linguistic Institute.

Ms. Litman will be the first woman graduate of the doctoral program of the Computer Science Department. Besides learning all about computer science, she has learned that weather does indeed matter. Finally, she has learned how to cook. Her speciality is homemade pasta.

Acknowledgments

I would like to thank my advisor, James Allen, for his guidance throughout this research as well as my graduate career. When I arrived at Rochester I knew very little about artificial intelligence and even less about doing research. James has been largely responsible for any improvements I might have made in either of these areas. I'd like to acknowledge his fine example, as well as his interest, time, and insights over the last five years.

I would also like to thank the other members of my thesis committee - Gary Dell, Jerry Feldman, and Pat Hayes - for their input and helpful comments throughout the various stages of this work. Thanks also to Pat for flying back for my defense, despite an upcoming "site visit from God and the Archangel Gabriel."

Much of this research developed from my work with the Knowledge Representation for Natural Language Understanding Group of Bolt Beranek and Newman, Inc. (aka BBN Labs). I would especially like to thank Candy Sidner for her interest, guidance, and many fruitful discussions. Most people feel lucky if they can find even one interested advisor - I feel especially fortunate to have had in effect two. I would also like to thank Brad Goodman and Marc Vilain. Their professional interest and comments, as well as their continued friendship and support, has meant a lot.

The academic environment at Rochester has been a particularly exciting yet pleasant one to be a part of. Thanks to the various members of the artificial intelligence community,

especially the gang of n and a.i. study group, for repeatedly listening to me speak and for reading my papers. I would particularly like to thank Henry Kautz for discussions and refereeing during all stages of this work. Thanks also to Brad Miller for reading the almost final draft, and to Mark Kahrs, Lee Moore (and the various hackers who had the bad luck to be around when I was) for their help with troff[1] and in general with preparation of this document.

One of the few bad things about student life is making then leaving so many good friends. Thanks to Paul Cooper, Gary Cottrell, Rick Floyd, Alan Frisch, Leo Hartman, Boris Heliotis, Elise Hill, Mark Kahrs, Henry Kautz, Jim Mayer, Lee Moore, Jill Orioli, Rich Pelavin, Don Perlis, Dan Russell, Lokendra Shastri, Dave Sher and Jay Weber, for lots of wonderful times.

Finally, to "m." for lots of 1+, food, and noises.

And, to my parents for their love and support (and the quals orchids).

This thesis is dedicated to the memory of Lydia Hrechanyk, who should have been the first.

[1] M. Kahrs and L. Moore, Adventures with Typesetter-Independent TROFF, Tech. Rep. 159, University of Rochester, June 1985.

Abstract

One promising computational approach to understanding dialogues has involved modeling the goals of the speakers in the domain of discourse. In general, these models work well as long as the topic follows the goal structure closely, but they have difficulty accounting for interrupting subdialogues such as clarifications and corrections. Furthermore, such models are typically unable to use many processing clues provided by the linguistic phenomena of the dialogues.

This dissertation presents a computational theory and partial implementation of a discourse level model of dialogue understanding. The theory extends and integrates plan-based and linguistic-based approaches to language processing, arguing that such a synthesis is needed to computationally handle many discourse level phenomena present in naturally occurring dialogues. The simple, fairly syntactic results of discourse analysis (for example, explanations of phenomena in terms of very local discourse contexts as well as correlations between syntactic devices and discourse function) will be input to the plan recognition system, while the more complex inferential processes relating utterances have been totally reformulated within a plan-based framework. Such an integration has led to a new model of plan recognition, one that constructs a hierarchy of domain and meta-plans via the process of constraint satisfaction. Furthermore, the processing of the plan recognizer is explicitly coordinated with a set of linguistic clues. The resulting framework handles a wide variety of difficult linguistic phenomena (for example, interruptions, fragmental and elliptical utterances,

and presence as well as absence of syntactic discourse clues), while maintaining the computational advantages of the plan-based approach. The implementation of the plan recognition aspects of this framework also addresses two difficult issues of knowledge representation inherent in any plan recognition task.

Table of Contents

List of Figures

Chapter 1

Introduction

1. Overview

Naturally occurring dialogues exhibit a wide variety of linguistic behavior problematic for existing natural language understanding systems. In particular, much of the interpretation process appears highly dependent on various types of discourse and pragmatic analyses. This work presents a computational theory and partial implementation of a discourse level model of dialogue understanding.

Consider the demands that the following dialogue (recorded at the information booth of a train station in Toronto (Horrigan [47])) would place on a computer system that could take the role of the clerk during the understanding process.

1) Passenger:	The eight-fifty to Montreal?
2) Clerk:	Eight-fifty to Montreal? Gate seven.
3) Passenger:	Where is it?
4) Clerk:	Down this way to your left. Second one on the left.
5) Passenger:	OK. Thank you.

Dialogue 1

In order to process initial sentence fragments such as utterance (1), such a system would need knowledge regarding a presupposed context of the dialogue. For example, the system could

use the knowledge that the speaker probably wants to either board or meet a train to infer that the speaker wants to know the train gate. Yet, since the dialogue is more than a series of unrelated question/answer exchanges, the system's understanding process is also dependent on the previous context provided by the dialogue itself. For example, the system could not understand the pronoun "it" and thus utterance (3) without using the knowledge provided by the preceding utterances. Furthermore, the system should be able to ascertain that utterance (3) initiates a clarification subdialogue with respect to the previous topic. Finally, the system should be able to exploit various linguistic signals used by the speaker, for example the use of "OK" to mark the conclusion of the clarification subdialogue and the use of "Thank you" to mark the conclusion of the dialogue.

Analysis of dialogues in other domains yields similar results. Consider Dialogue 2 (Sidner and Bates [89]), the initial portion of a scenario developed from protocols in which a user interacts with an editing system to manipulate network structures in a knowledge representation language.

6) User:	Show me the generic concept called "employee."
7) System:	OK. <system displays network>
8) User:	I can't fit a new IC below it. Can you move it up?
9) System:	Yes. <system displays network>
10)User:	OK, now make an individual employee concept whose first name is "Sam" and whose last name is "Jones." The Social Security number is 234-56-7899.
11) System:	OK.

Dialogue 2

As above, the system will need to be able to relate current utterances to previous utterances in the dialogue. In particular, the system must recognize that some utterances introduce and continue a topic (e.g. the subdialogues corresponding to execution of an editing plan in lines (6)-(7) and (10)-(11)) while others temporarily interrupt a topic (e.g. the correction subdialogue

corresponding to lines (8)-(9)). Furthermore, relationships between utterances occur not only across but also within subdialogues, as in lines (8) and (10). And again, we see the use of linguistic clues such as "OK" and "now" to explicitly signal utterance relationships (for example, in (10) the resumption and continuation of the interrupted topic corresponding to execution of the editing plan).

Finally, imagine a system capable of taking the role of the operator and clerk in Dialogue fragments 3 and 4, respectively.

12) User: Could you mount a magtape for me? It's tape1. No ring please. Can you do it in five minutes?

13) Operator: We are not allowed to mount that magtape. You'll have to talk to operator about it. After nine a.m. Monday through Friday.

14) User: How about tape2?

Dialogue 3

15) Passenger: Trains going from here to Ottawa?

16) Clerk: Ottawa. Next one is at four-thirty.

17) Passenger: How about Wednesday?

18) Clerk: One at nine thirty, nine thirty in the morning, four thirty in the afternoon...yeah, that's it.

Dialogue 4

(Dialogue 3 is a terminal transcript of a user/operator link, provided by Bill Mann. Dialogue 4 is from the same corpus as Dialogue 1). As in the above dialogues, such a system will need to be able to recognize various relationships between utterances. For example, the latter portions of line (12) elaborate an initial request to mount a tape, while line (14) modifies and replaces the whole set of utterances corresponding to the elaborated request. Furthermore, in both dialogues we see the use of "how about" as an explicit signal for the modification relationship. As in Dialogue 1, the system will also need to use some sort of context to understand sentence fragments and more generally elliptical utterances. While the linguistic context of the previous discourse is sufficient to understand (14), to understand (15) and (17) knowledge regarding

some other context is also needed. In line (15) this is because there is no previous discourse to draw upon, while in line (17) this is because the concept that Wednesday replaces is only implictly part of the preceding dialogue.

This dissertation will present a computational theory and partial implementation of the dialogue understanding process that addresses these issues. As we will see, the theory extends plan-based approaches for sentence and simple dialogue understanding by incorporating more linguistic-based insights from the area of discourse analysis. In particular, the simple, more syntactic results of discourse analysis (for example, explanations of phenomena in terms of very local discourse contexts as well as correlations between syntactic devices and discourse function) will be used without change, while the more complex inferential processes relating utterances have been totally reformulated within a plan-based framework. Such a theory will enable the handling of a wide range of linguistic phenomena while maintaining the computational advantages and complementary coverage of the plan-based approach. Such a model recognizes that a dialogue can (and for robustness as well as efficiency, should) be analyzed along several dimensions.

2. The Discourses and their Analyses

The theory presented in this work draws upon the analysis of four sets of person-person dialogues, characterizing situations in which the desirability of a computer system as a conversational participant is easily imagined. Examination of such dialogues provides data regarding the kinds of language people will likely use in similar person-machine interactions as well as indications of the kinds of interpretations people construct from such utterances.

Although the dialogues are somewhat restricted in topic and involve only cooperative exchanges, we will see that they nonetheless exhibit many interesting linguistic phenomena characteristic of more freewheeling exchanges. They thus provide a nice testbed for developing a computationally tractable system that yet addresses some complex linguistic issues.

Before discussing the data, a bit of terminology will be useful. A *discourse* will refer to any exchange involving more than a single sentence, for example texts, paragraphs, dialogues, conversations, and stories. An *utterance* will refer to both a speaker's complete turn as well as individual sentences within a turn; the intended usage should always be clear from the context. For example, in the context of Dialogue 2 line (8) will be referred to as the user's second utterance, while in the context of line (8) "Can you move it up?" will be referred to as the user's second utterance. Finally, since the model treats spoken texts as written texts (i.e., intonation cues, etc. are ignored) the terms *speaker, hearer,* and *utterance* will be used loosely for all the dialogues.

2.1. The Data

2.1.1. Train Station Information Clerk

With the permission of the station master, Horrigan [47] tape recorded a corpus of dialogues between people seeking information and a clerk in the "green light" booth in Union Station, Toronto, July 1976. Of the four hours of dialogue collected approximately the first hundred dialogues were transcribed, seventeen of which looked especially interesting and were selected for further analysis.

The dialogues are examples of *information-seeking dialogues*, dialogues in which an agent seeks information with respect to a plan that will not be executed during the dialogue. They are similar to both the question-answering dialogues of Grosz [37], where a person queried a (simulated) data base in order to solve an assigned problem, and the information seeking dialogues of Carberry [15] in the domain of university courses, policies, and requirements.

The advantages of the train corpus were numerous. The dialogues were collected in a totally natural setting, yet provided data on the type of language people would use if they had verbal access to an intelligent provider of information. Furthermore, the dialogues were sim-

ple. Since they were typically short in length (less than a dozen lines) the theory could easily be tested on full dialogues as opposed to just fragments. Similarly, since they were limited in topic a small set of underlying plans formed the basis for a large number of dialogues. Practically all of the dialogues contained questions only related to meeting trains, boarding trains, or locating rooms or offices in the train station. Yet, despite their simplicity the dialogues exhibited the problematic linguistic phenomena targeted for investigation in this research.

2.1.2. KLONE-ED system

Sidner [87] collected a set of eight protocols, each approximately two hours in length (i.e. several single-spaced pages), between simulated natural language understanding systems and users manipulating a database using natural language and a graphics display. Three of these protocols simulated KLONE-ED, a graphic editing system that could manipulate structures in the domain of the knowledge representation language KL-ONE [11]. To collect these protocols, users were given a specific task to perform using the KLONE-ED system. All interactions took place via a computer terminal using natural (i.e. non-simplified) English, and users were aware that a person was simulating the KLONE-Ed system. From these protocols, Sidner and Bates [89] constructed a prototypical scenario containing a subset of the capacities possessed by the simulated systems, concentrating on those that seemed plausible for the near future. Dialogue 2 is the initial portion of this scenario.

The KLONE-ED dialogues are examples of *task-oriented dialogues*, dialogues in which agents work cooperatively on a task that is performed during (and via) the dialogue. However, the dialogues are unusual in allowing both linguistic and graphic modes of interaction. In particular, the KLONE-ED system's interaction with the user is often non-linguistic, with utterances only being produced to satisfy simple conversational conventions.

2.1.3. Computer Operator

The third corpus of dialogues was provided by Bill Mann and consists of sixteen (cleaned-up) computer terminal transcripts collected when users linked to a (human) operator. Like the train dialogues, the set of tape dialogues were collected in natural, as opposed to experimental, situations. Like in the KLONE-ED dialogues, the mode of communication was the more typical typed (rather than spoken) computer interface. In general the lengths of the transcripts are approximately one type written page.

This corpus contains both information-seeking and task-oriented dialogues. For example, there are dialogues where the user only wants a question answered, such as:

Linker: Do you know if system will really be up all night?

Operator: Unless we crash!

There are also task-oriented dialogues such as Dialogue 3, where the user and the operator need to cooperate in order to perform the user's task. Finally, the corpus contains some dialogues that exhibit examples of both types of interactions.

2.1.4. Chinese Cooking Consultant

The final corpus of dialogues was collected by Kahrs et al [50]. The data consists of two computer terminal transcripts in which a (human) expert guides a novice in the preparation of a Chinese meal. As in the KLONE-ED transcripts the dialogues correspond to several pages of single spaced text and were elicited solely for the purposes of data collection. As in the task-oriented dialogues of Grosz [37], where an expert instructed an apprentice on the assembly of part of an air compressor, collection of the dialogues provided data on the language requirements of a possible computer consultant. Unfortunately, some of the data collected from these dialogues was a bit too interesting with respect to the scope of this research. For example, the fact that in both dialogues the expert and novice were friends led to the discussion of too many topics unrelated to the immediate cooking task at hand.

2.2. The Data Analysis

Despite the differences among the sets of dialogues with respect to domain, genre, mode of communication, length, spontaneity, and so on, a number of discourse level phenomena are characteristic of all the interactions. Such generalities provide a natural set of goals for the design of systems that ultimately will be able to understand the type of unrestricted English now used in corresponding person-person interactions. The following sections will present the results targeted for this particular research, identifying both the linguistic phenomena and their implications for the design of a natural language system. (While the data could also be analyzed to see how the frequency of such phenomena varies depending on such features as mode of communication or gender of speaker, those types of results are irrelevant for the purposes of this particular research).

2.2.1. Subdialogues and their Relationships

Any extended dialogue can be decomposed into *subdialogues*, cohesive subunits that can themselves be decomposed into further subdialogues. Grosz [37] noted that removal of such subunits does not seem to effect the coherency of the larger unit. She also noted various linguistic devices supporting this segmentation phenomena.

The range of subdialogues exhibited varies across the sets of data. For example, in the task-oriented exchanges subdialogues correspond not only to execution of the subtasks but also to meta-discussions such as clarifications and corrections of the subtask execution. Recall Dialogue 2. Lines (6)-(7) and (10)-(11) are subdialogues corresponding to editing subtasks, while (8)-(9) is an interrupting subdialogue correcting the execution results of the previous subtask.

In the more personal cooking dialogues, subdialogues totally irrelevant to execution of the task at hand occur, for example the gringo exchange of the following fragment:

M: Are you going to use the wok?

L: Yes, of course! How else does one cook szechuan food?

M: OK, then you should slice the garlic. Use 1-2 cloves. There are gringos in the world my dear...

L: That's irrelevant. We're cooking chinese, not mexican food, senor.

M: There are gringos in EVERY cuisine!! i.e., there exists an x s.t. x is a cook an x is mapped onto a prototypical gringo.

L: Do you know the chinese equivalent of gringohood?

M: No, I'm afraid I'm speechless.

L: Oh well, I don't know either. Back to the kitchen.

In contrast, subdialogues in the information-seeking exchanges do not correspond to execution of subtasks, since such execution takes place outside the dialogue. Recall Dialogue 1, where the user's underlying goal is to board the eight-fifty to Montreal. Lines (1) and (2) form a clarification subdialogue regarding the departure gate of the train to be boarded, while lines (3)-(4) form a clarification of the previous clarification.

Finally, the train station corpus has a large number of subdialogues corresponding to communication checks, due to the noisy environment. For example, consider the last two utterances of the following fragment:

P: Going to Stratford, what gate would it be?

C: Which one is that?

P: Two fifteen, I think is the ...

C: Yeah, two fifteen. Gate number eight.

P: Number eight?

C: Right.

Not only do the subdialogues differ in content, but they also differ in the way they relate to the existing discourse context. For example, Grosz [37] noted that in task-oriented dialogues subdialogues corresponding to execution of subtasks could be related to one another via the corresponding subtask execution structure. In other words, subtasks are generally related via a

hierarchical tree structure; discussion of such subtasks generally precedes depth-first through this tree. Thus, the backbone of the Chinese cooking dialogues should (and does) consist of subdialogues organized according to the execution of the stages of the recipe.

Unfortunately the other types of subdialogues do not appear to fit into this framework. In particular, many non-subtask subdialogues suspend, rather than continue, traversal of such task structures; the subtask traversal is then resumed when the interrupting subdialogues are concluded. For example, in Dialogue 2 the correction exchange of lines (8)-(9) temporarily interrupts the flow of subtask subdialogues. Such an interruption dynamically occurred due to the unanticipated aspects of the system's network display. In information-seeking dialogues such as Dialogue 1 where the task execution is non-linguistic, such interruptions are all that appear linguistically; use of the information booth is unnecessary when plan execution goes smoothly. While the interrupting subdialogues are not generally organized into some sort of larger global structure, they often can be related to the subdialogue interrupted. For example, recall the clarification and correction relationships in Dialogues 1 and 2 with the preceding subdialogues (lines (3)-(4) and (8)-(9), respectively). However, as illustrated by the cooking exchange relationships that further the achievement of the underlying task are not necessarily present.

The implications of these observations for the design of a natural language understanding system are many. For example, the existence of subdialogues as cohesive units requires a system that can recognize the boundaries of such units. Furthermore, a system should be able to discriminate between certain kinds of subdialogues, since subdialogues corresponding to subtasks, meta-discussions of subtasks, and interruptions unconnected to subtasks relate to the previous discourse in different ways. For example, depending on the context some relationships between subdialogues are more expected than others. Furthermore, the interpretation of utterances depends on the relationship inferred.

The data analysis also indicates what kinds of knowledge an intelligent computer system will need to understand such dialogues. As Grosz [37] noted, recognition of a dialogue's discourse structure is necessary to explain a class of linguistic phenomena. Since the structure of task subdialogues corresponds to the execution structure of corresponding subtasks, knowledge regarding the structure of typical domain tasks is necessary. However, since in many of the dialogues agents do more than merely execute a plan, knowledge about higher level processes such as plan debugging will be useful. Finally, since any subdialogue may be temporarily interrupted, structures for managing interruptions are necessary.

2.2.2. Sentence Fragments and Elliptical Utterances

Each corpus of dialogues also contained sentence fragments or other elliptical utterances. While in isolation such utterances containing missing words or phrases are syntactically incomplete, in the context of a discourse the missing entities can usually be recovered. For example, in Dialogue 3 the missing portions of line (14) can be recovered using line (12). Thus, our desired natural language understanding system should be able to both maintain and use portions of the previous dialogue.

Unfortunately, many types of elliptical utterances cannot be handled using only the content of the previous dialogue. Consider analysis of utterances (1) and (15), the initial fragments of Dialogues (1) and (4), respectively. Since there is no preceding dialogue, to find the missing phrases the system will need to draw upon an extra-linguistic context of knowledge about the world and likely goals of the speaker. In other words, if the train clerk knows that persons seeking information typically are boarding a train, meeting a train, or looking for a room in the station, "The eight-fifty to Montreal" can be understood by using these plans to provide the missing information (in this case, knowing that to board a train an agent needs to know what gate to go to). Such analysis is also useful for understanding non-elliptical utterances. Since the system not only knows what was said but also why, recognition of how an utterance con-

necting with a speaker's underlying goal provides a deeper level of understanding.

While the use of planning knowledge in understanding utterances (1) and (15) (or any utterance in isolation) is not a discourse level phenomena, it is similar in that understanding involves connecting an utterance to some previous context, in this case an extra-linguistic context. Thus, connecting an utterance to a context of speaker goals can usually provide an alternative to solely linguistic explanations of discourse level ellipsis. This suggests that a natural language system should be able to use and coordinate both linguistic and plan-based analyses of the same phenomena. Furthermore, plan-based analyses appear to be able to explain discourse level examples that are problematic for the linguistic method. Recall "How about Wednesday" in Dialogue (4), where the entity that "Wednesday" replaces is not explicitly mentioned in the previous dialogue.

2.2.3. Surface Linguistic Phenomena

Many researchers have noted that *surface linguistic phenomena*, e.g. the particular lexical items and syntactic structures used in an utterance, explicitly signal the role of an utterance with respect to the overall discourse. It has been shown that choice of referring expressions (for example pronouns and definite noun phrases) varies depending on the status of the subdialogue containing the entity. Also, many seemingly insignificant words and phrases not only mark transitions between subdialogues, but also indicate the relationships between such subdialogues. For example, consider the use of "OK" in Dialogues 1 and 2, "how about" in Dialogues 3 and 4, and "by the way" in the following dialogue fragment.

L: The eggplant has been sliced. It's (good) that you advised cutting by judgement instead of absolute directions. We got a monster eggplant that split into ten sections. *By the way*, the eggplant is turning brown. The traditional method for preventing oxidation is to sprinkle the food with lemon juice. Do you recommend doing so?

M: I'm not sure that it's necessary since we're going to use it soon. If you would like to, you can, but the lemon taste may carry over.

L: I dig. Well skip it then.

"By the way" typically indicates not only the beginning of a new subdialogue , but also that the subdialogue temporarily interrupts the subdialogue structure corresponding to the preparation of the eggplant recipe as guided by the consultant. With respect to the design of a natural language system, these results indicate that systems should be able to use such discourse clues. However, since such clues are not always present (for example, no clue precedes the interrupting utterance (8) in Dialogue 2), systems should also be able to proceed as best they can without them. Thus, we again see the need for linguistic as well as extra-linguistic knowledge.

2.2.4. Multi-Sentential Utterances

Often a speaker will express a single thought via a set of utterances rather than a single utterance. For example, the first three utterances of Dialogue 3 could easily have been replaced with the single utterance "Could you mount tape1 for me with no ring please?" Just as subdialogues needed to be related to one another in various ways, sentences in multi-sentential utterances (i.e. sentences within a subdialogue) will need to be related. For example, to understand "It's tape1" in Dialogue 3 a system will need to be able to relate it to the previous discourse and/or plan context provided by "Could you mount a magtape for me?" A desirable system design would be one that uses the same structures and mechanisms to recognize the previously discussed relationships.

3. A Computational Theory of Dialogue Understanding

As mentioned above, the results of the discourse analysis along with constraints of computational plausibility both guided and later supported the design of the theory presented in this dissertation. Previous work in the area had generally concentrated on either (1) the computational recognition and use of plan structures for understanding single sentences and dialogues without interruptions, or (2) linguistic explanations of interruptions and other discourse phenomena often dependent on computational processes that were unrealistically presupposed. In contrast, this work was based on the desire to use both planning and linguistic knowledge

during the process of dialogue understanding [4, 56]. The result was an investigation that concerned itself primarily with extension of the results of the plan-based work using issues and tractable results of the more linguistic work.

The initial phase of this research [57] focused on the clarification subdialogues and surface phenomena of the train domain, and resulted in the hypothesis that an important kind of interrupting relationship between utterances could be recognized by extending the system's knowledge regarding specific plans to include knowledge about things people could do with such plans (e.g. introduce them, execute them, clarify them, debug them...). Yet, as in the linguistic theories, surface phenomena could still be used to guide the recognition of underlying dialogue (here plan) structures, and conversely recognized structures could be used to explain other surface phenomena. In other words, a way of coordinating alternative linguistic and plan-based explanations of the examined set of discourse phenomena was developed. The initial theory was then applied to the other dialogues (for example, see [58]). While some of the particular details needed to be generalized, the basic structures and algorithms held. In particular, the scope of the theory was extended to include dialogues with (and without) several kinds of interruptions. Also, it turned out that ellipsis and the (very limited) number of multi-sentential utterances could be analyzed without recourse to any new theoretical mechanisms. The final result was the formal theory and partial implementation of a system capable of understanding a set of discourse level phenomena in many dialogue variants.

Figure 1.1 presents the components and interactions that would be found in such a system. For every utterance, the system's processing would consist of both linguistic and plan-based analyses of discourse level phenomena. In other words, the design of the system is based on two major assumptions:

(1) People form and execute plans containing linguistic (as well as non-linguistic) actions, plans that other agents can infer from observation of these actions.

(2) The structure of dialogues and the use of surface linguistic phenomena are highly rule-governed.

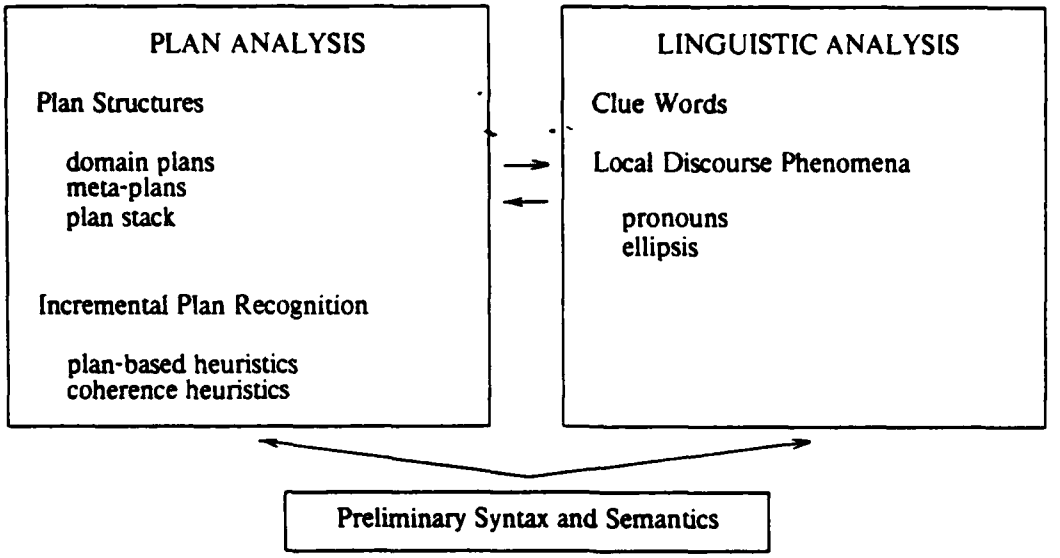


Figure 1.1: System Overview

The scope of the work is constrained by also assuming that agents will cooperate and share the same knowledge.

The input to such a system is a syntactic and semantic analysis, typical of the kind of output produced by existing parsing systems. As the dialogue progresses, the system performs the necessary plan analysis (formalized in the incremental plan recognition algorithm) using knowledge about the structure of typical plans (domain plans), a preliminary set of things people do with plans (meta-plans), and a previous dialogue context consisting of the set of previously recognized executing and interrupted plans and their relationships (maintained via the plan stack). The recognition algorithm is a search process guided by both rules of rational planning behavior and rules about the structure of the dialogues in terms of relationships between the higher-level (meta) planning processes that underly them. Finally, clue words present in the utterance as well as typical linguistic analyses of local discourse phenomena are also input to the plan recognizer and used to constrain its default plan-based search process. Note, however, that if such linguistic analyses are not available, plan recognition can still

proceed, and in fact can be used to provide alternative plan-based analyses of the local linguistic phenomena.

As we will see, the behavior just described is fully specified by the theory. The implementation, however, is only partial and corresponds to the major contribution (i.e. the plan recognition aspects) of the theory. In particular, the current implementation illustrates the recognition of a stack of executing and interrupted domain and meta-plans from a representation of the parse of "The eight-fifty to Montreal?" The implementation also addresses two difficult issues of knowledge representation (related to equality reasoning) inherent in any plan recognition task. Finally, while the parser and linguistic analyses are simulated in the current system, actual implementations demonstrating such capabilities do exist either at Rochester or other institutions.

Figure 1.2 illustrates (at a very high-level) how the theory is used to simulate a system processing a dialogue by constructing and manipulating a stack of recognized user and system plans and meta-plans. For example, given an input such as (the syntactic and semantic analysis of) the noun phrase "The 8:50 to Montreal," the system, here taking the role of the clerk, will use its knowledge regarding domain plans, meta-plans, the previous discourse (currently an empty stack), and the plan recognition algorithm to hypothesize that the passenger is introducing a plan for the system to clarify the passenger's domain plan to board the 8:50 to Montreal. The system performs this analysis by recognizing the various plans and their relationships, then placing them on the stack. Each meta-plan on the stack refers to the plan below it, with the domain dependent task plan at the bottom. The top plan is currently executing and the others will be resumed when the plan immediately above is popped. The system will then manipulate this stack and generate an appropriate response; the resulting stack provides a context for understanding the next user utterance. To understand "Where is it?" the system will, as before, analyze the passenger's utterance as an introduction of a plan for the system to clarify a plan. However, in this case the clarification is with respect to the previously executed system

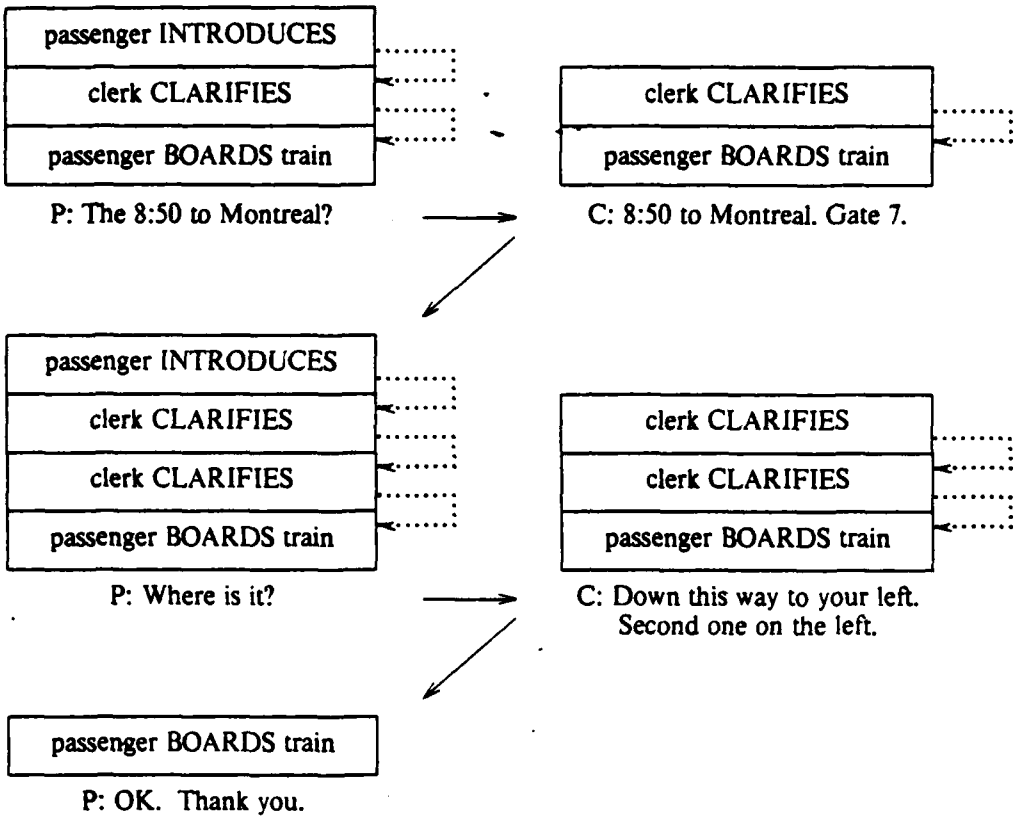


Figure 1.2: Example

clarification rather than a new passenger domain plan. In other words, once the dialogue is in progress the system prefers an interpretation that coheres with the previous dialogue. Finally, understanding of the last passenger utterance shows how the system can use linguistic clues to guide its default (plan-based) manipulations of the plan stack.

4. Dissertation Overview

The next two chapters will present the structures and algorithms of the theory in detail. Chapter 2 will present the plan-based aspects, while Chapter 3 will show how various insights in the area of discourse analysis have either been reformulated in or interfaced to this frame-

work.

Chapters 4 and 5 will show how the theory can actually be used to process the four examples given at the beginning of this chapter. Chapter 4 will concentrate on the recognition of interrupting subdialogues (both clarification and corrections), while Chapter 5 will concentrate on the use of another type of interrupting subdialogue to explain sentence fragments, linguistic, and extra-linguistic elliptical utterances. While the examples are quite detailed and thus somewhat tedious, to fully understand the theory the reader should at least comprehend the first example of Chapter 4.

Chapter 6 will discuss technical issues relating to the implementation. In particular, the chapter will concentrate on the discussion of two modes of reasoning that many current knowledge representation systems lack but all plan recognition systems need. The chapter will show how the current implementation of the plan recognition process for "The eight-fifty to Montreal?" addresses these (as well as the plan recognition) issues. For example, the chapter will show how the process of constraint satisfaction can be used to implement the recognition of object plans from meta-plans. Finally, the chapter will discuss the significance of the implementation with respect to the theory. We will see that unlike many discourse models, all computational processes required by the theory have either been implemented in this system or simulated here and implemented elsewhere. Chapter 6 can be skipped by readers not concerned with either issues of implementation or knowledge representation.

Finally, Chapter 7 will place this work into the context of the relevant literature, while Chapter 8 will summarize and elaborate on future directions.

Chapter 2

Plan Analysis

1. Background

Plans, sequences of actions that achieve a set of goals, are a central concept in artificial intelligence research. Early work was in the context of robot problem solving systems (Newell and Simon [67], Fikes and Nilsson [30]) and involved generating plans, linear sequences of executable robot actions, given an initial world state, a goal state, and a library of actions a robot could perform. Actions were formally modeled as operators that changed one state of the world into another; mechanisms were developed for searching through state spaces to find operator sequences connecting initial and goal states. Later work extended the framework to include hierarchical (Sacerdoti [79, 80]) and non-linear (Sacerdoti [80], Tate [92]) planning. By allowing plans to be developed level by level, i.e. hierarchically, low level details could be postponed. Plans at each level were thus shorter and more manageable. This work was also in the context of a robot generating plans for a toy "blocks world" domain.

Many artificial intelligence researchers have used such robot planning frameworks to address a wide variety of issues in the area of natural language. Bruce [14] suggested and Allen, Cohen, and Perrault [3, 22] pursued a plan-based approach to conversation based on insights from the philosophy of language (Austin [9], Searle [84], Grice [35]). The work in philosophy suggested viewing utterances as *speech acts*, actions performed by speakers to achieve

intended effects. Understanding an utterance thus involved both constructing a literal interpretation as well as recognizing underlying intentions. Allen, Cohen and Perrault, adopting this purposeful view of language, developed computational models for recognizing and generating speech acts based heavily upon the work in robot problem solving. For example, speech acts were modeled as action operators in a language planning system; understanding a speech act involved recognizing the speakers intentions (i.e. plan). Their theory produced a new view of question-answering conversations, and led to systems that could provide more information than required as well as understand indirect speech acts and sentence fragments.

Other plan-based work has been concerned with issues of discourse context, i.e. relating the current utterance to the previous utterances in the conversation. For example, goal analysis has been used to relate sentences processed in story-understanding systems. Schank [82] noted that stereotypical stories could be understood by the use of a script, a data structure for representing such stereotypical situations. Scripts provided expectations in the form of slots, which were filled in during story understanding. Wilensky [96] generalized this idea by using a more flexible intentional (i.e. plan-based) analysis. By reasoning about the story situations in terms of interacting goals and plans of the characters, his system could understand novel (as well as stereotypical) goal-based stories. With respect to conversational analysis, Grosz [37] noted that in task-oriented dialogues the topic usually follows the task (i.e. plan) structure. She used this result to process various linguistic phenomena exhibited in the dialogues, such as definite noun phrases and elliptical utterances. Sidner and Israel [86] and Carberry [15] have used similar intuitions to recognize multi-step plans. Carberry used the plan context to help track the changing task goals of a speaker during information-seeking dialogues. Similarly, Sidner and Israel extended Allen [3] by using plan knowledge to provide a context. They also suggested using the framework to recognize interruptions of faulty plans. This problem of when to ignore the expectations provided by discourse context, as in an interruption or change of plan, has generally been ignored. Sidner [90] begins to tackle the problem of interruptions

in order to recognize when two or more plans underlie a discourse. Related to this issue is how the actual phrasing of the utterance controls (whether to overrule or reinforce) the plan recognition process. As will be seen in the next chapter, issues of interruptions and linguistic analysis have been of concern primarily outside the plan-based field.

This chapter will present a new theory of plan recognition, one that will be able to systematically use (or ignore) the previous conversational context and thus handle a wide variety of subdialogues. As will be seen, in addition to the standard domain-dependent knowledge of task plans, some knowledge about the planning process itself will be introduced. This will be done via *meta-plans*, domain-independent plans that refer to the state of other plans. During a dialogue, the theory will specify how to incrementally recognize instantiations of such plans and put them on a stack, each meta-plan on the stack referring to the plan below it, with the domain-dependent task plan at the bottom. In the next chapter we will see that the manipulation of this stack of plans is similar to the manipulation of topic hierarchies that arise in discourse models. In that chapter we will also see how the plan recognizer can use some of the results of the more linguistic models.

2. Plan Structures

2.1. Models of Plans

In a plan-based approach to language understanding, an utterance is considered understood when it is related to some underlying plan of the speaker. The hearer must thus bring to the understanding task some knowledge about typical speaker plans. A library of *plan schemas* will be used to represent this type of general knowledge. (*Plan instantiations* are formed from such general schemas by giving values to the schema parameters.)

Plan schemas can be used for both plan generation and plan recognition. For example, a planning system would use these schemas in the same way it would have used STRIPS action

descriptions [30], i.e., to generate sequences of matched and instantiated schemas to achieve some goal. Once generated, the complex plan instantiation is executed much as one would run a program. A plan recognizer, on the other hand, will use the plan schemas to recognize the plan instantiation that produced an executed action. In particular, the recognizer will be concerned with recognizing plan instantiations from actions executed as part of a dialogue. *Plan* will be used loosely to refer to both plan schemas and plan instantiations. The intended meaning should always be clear from the context.

Every plan has a *header*, a parameterized action description that names the plan. *The parameters of a plan* are the parameters in the header. As usual in many models of planning (for example, STRIPS [30]), action descriptions are represented as operators on the planner's world model and are defined in terms of *prerequisites* and *effects*. Prerequisites are conditions that need to hold (or be made to hold) in the world model before the action operator can actually be applied. Effects are statements that are asserted into the world model after the action has been successfully executed. By assuming that all other aspects of the planner's world model remain unchanged, the frame problem [63] can be suppressed. Since the particular plans that will be used in this work have prerequisites that aren't falsified, and so on, it will not be necessary to have effects that delete statements from the world model, as in STRIPS.

Action descriptions may also have *decompositions*, which enable hierarchical planning (Sacerdoti [80]). Although the action description of the header may be usefully thought of at one level of abstraction as a single action achieving a goal, such an action might not be executable, i.e. it might be an *abstract* as opposed to *primitive* action. Abstract actions are in actuality composed of primitive actions and possibly other abstract action descriptions (i.e., other plans). Thus, decompositions may be sequences of primitive actions, abstract actions, goals to be achieved (action sequences to be dynamically constructed) or a mixture. Note that the usual distinctions between the terms "action" and "plan" have become blurred. In STRIPS, (primitive) actions were organized into plans. Here, as in ABSTRIPS [79] and NOAH [80], plans

(abstract or primitive actions) are organized into larger plans. It is useful to precompile well-defined plans for a goal as abstract actions to capture their generality as components within higher level plans.

A few things atypical of such planning models should also be noted. Associated with each plan is a set of *constraints*. These are similar to prerequisites, except that the planner never attempts to achieve a constraint if it is false. Thus, any action whose constraints are not satisfied in some context will not be applicable in that context.[1] Also, plans may involve both physical and linguistic actions. In a typical plan involving a conversation, for example, agents often take turns executing actions corresponding to the utterances in the conversation. Finally, plans may contain actions with both the system and the user as possible agents. Since both agents are assumed to be cooperating, there is no reason why the user can't construct a plan that depends on the system's help.

Figure 2.1 illustrates a very simple plan schema with header "BUY-TICKET (passenger, clerk, ticket)" and parameters "passenger," "clerk" and "ticket." An instantiation of such a plan schema might be generated and then executed to change a world in which a passenger doesn't have a ticket into one where he or she does. Before the plan instantiation can be performed, the prerequisites indicate that the passenger must have (or construct a subplan to obtain) enough money to pay for a ticket. Similarly, the constraints indicate that the clerk must be a ticket-seller. However, unlike a prerequisite, if this condition is not already true then the ticket cannot be bought from this particular clerk. In other words, it doesn't make sense to treat this condition as a goal to be achieved since it is beyond the passenger's capabilities. Assuming that these conditions are met, buying a ticket can then be performed by having the passenger first pay the clerk, followed by the clerk giving the passenger the ticket. The world model is then updated to indicate that the passenger now has the ticket and the clerk

[1] These constraints should not be confused with the constraints of Stefik [91], which are dynamically formulated during hierarchical plan generation and represent the interactions between subproblems.

now has the money.

HEADER:	BUY-TICKET(passenger, clerk, ticket)
PREREQUISITE:	HAS(passenger, price(ticket))
DECOMPOSITION:	PAY(passenger, price(ticket)) GIVE(clerk, passenger, ticket)
EFFECTS:	HAS(passenger, ticket) HAS(clerk, price(ticket))
CONSTRAINT:	TICKET-SELLER(clerk)

Figure 2.1: A Plan Schema

When the implementation is presented, we shall see how such schemas are axiomatized using a typed horn clause logic [6], where types are organized into hierarchies as commonly found in semantic network formalisms. The naming of the parameters in the schemas will reflect such type restrictions. Thus, when an instance of the schema in Figure 2.1 is created, *passenger* and *clerk* are restricted to people (or systems) and *ticket* is restricted to tickets. We will also see how the representation of plan schemas as described in this chapter glosses over difficult issues of knowledge representation, and thus does not exactly correspond to the representation of schemas in the implementation. Other assumptions that underlie the above representation follow. In general, relaxation of any of the assumptions produces a topic worthy of its own research effort.

Although a STRIPS based plan representation is typical, there are ultimately several limitations. For example, the only temporal constraint on a plan schema is an implicit linear ordering of the actions in the decomposition. Thus, in Figure 2.1, the clerk must be paid before the ticket is given. In NOAH [80] a representation enabling non-linear planning was developed. More recently, Allen [7] has developed a much more general theory of action and

time useful for a plan generation system (Allen and Koomen [5]). Regardless of the time issue, a STRIPS based representation system will also prove inadequate for certain types of plan inference. Pollack [73] is developing a more expressive model that will enable reasoning about plans that an agent might have even though they are unrealizable.

All issues involving non-mutual beliefs of agents will be ignored. In other words, it is assumed that what agent A believes that agent B believes is equivalent to what agent B believes. For example, all participants in the dialogue share the same plan library; knowledge about what plans exist as well as how they are performed is mutually believed by all. While greatly simplifying representation issues, this assumption would need to be relaxed when understanding dialogues containing deceit (Bruce [13]) or various types of miscommunication (as discussed in Chapter 8).

2.1.1. Domain Plan Schemas

Domain plan schemas represent typical tasks that might be performed in a given domain. Such knowledge has been the mainstay of previous plan-based works. Figure 2.2 presents a subset of the relevant domain plan schemas necessary for processing the dialogues in the Toronto train station. Although the formalizations are obviously incomplete, they will be sufficient for the purposes of this dissertation.

Since, as mentioned above, the naming conventions in the figures presume an underlying type hierarchy, it will be useful to briefly discuss the particular type hierarchy used before discussing the figures in depth. Obviously one of the types needed will be a train type, defined as follows:

```
(subtype TrainType PhysicalObjectType
        (gate LocationType)
        (station CityType)
        (time TimeType))
```

Note that the representation permits complex structured types, i.e. frame-like structures [66],

HEADER:	GOTO(agent, location, time)
EFFECT:	AT(agent, location, time)
HEADER:	MEET(agent, arriveTrain)
DECOMPOSITION:	GOTO(agent, gate(arriveTrain), time(arriveTrain))
HEADER:	BOARD(agent, departTrain)
DECOMPOSITION:	GOTO(agent, gate(departTrain), time(departTrain)) GETON(agent, departTrain)
HEADER:	TAKE-TRAIN-TRIP(agent, departTrain, destination)
DECOMPOSITION:	SELECT-TRAIN(agent, departTrain, departTrainSet) BUY-TICKET(agent, clerk, ticket) BOARD(agent, departTrain)
CONSTRAINTS:	EQUAL(destination, station(departTrain)) EQUAL(destination, station(departTrainSet)) EQUAL(departTrain, object(ticket))

Figure 2.2: Domain Plan Schemas for the Train Domain

by allowing a set of distinguished function names called *roles*, here "gate," "station" and "time." The role values are type restricted and can be accessed functionally. Thus "gate(train1)," where train1 is an instance of type TrainType, refers to the specific gate filling the gate role.

Trains will be further decomposed into arriving and departing trains. More formally,

(subtype ArriveTrainType TrainType)
(subtype DepartTrainType TrainType)

As common in semantic network hierarchies, subtypes inherit the properties of their super-types; thus, these two types inherit the three roles of TrainType. Finally, implicit in these types is the fact that the information booth is in Toronto. It is assumed that gates and times refer to Toronto, while the station refers to the other city that the train is either going to or coming from.

The first plan in Figure 2.2 summarizes a simple plan schema with header "GOTO(agent,location,time)," with parameters "agent," "location," and "time," and with the effect "AT(agent,location,time)." The header specifies a primitive action so there is no decomposition. The prerequisites and constraints are not shown. Throughout the dissertation, only the parts of the plan schemas needed for the examples will be presented.

The second plan summarizes a plan schema for the abstract action MEET, which is executed by performing a constrained version of the primitive action GOTO. As mentioned above, constraining the type of train to be met to arriving trains, defined as trains going to Toronto, captures the knowledge that the information booth, and hence the agents, are in the Toronto station. Thus this plan schema would best be described in English as "meeting a train at a gate in the railroad station of Toronto" rather than as the general action of "meeting a train."

The BOARD plan schema is similar to MEET, and is one step of the complex plan schema TAKE-TRAIN-TRIP (again, implicitly from Toronto). The first constraint of TAKE-TRAIN-TRIP captures the fact that the train taken, i.e. *departTrain*, must have as the value of its station role *destination*. The second constraint indicates that this is also the only restriction on the the set of possible candidates for *departTrain* used by SELECT-TRAIN. The third constraint indicates that the ticket purchased will be used to take *departTrain*. The specification of the other plans needed in this domain, e.g. plans to select a particular train from a set of possibilities, plans to buy tickets, plans to ask directions, etc., are not needed to process the examples chosen.

Since domain plans are domain dependent, participation in the KL-ONE or tape dialogues involves reloading the initial plan library with an appropriate set of domain plans. The actual plan schemas used in those domains will be given with the examples in later chapters.

2.1.2. Meta-Plan Schemas

Plans about plans, or *meta-plans*, deal with introducing plans, executing plans, specifying parts of plans, debugging plans, abandoning plans, etc., independently of any domain. Although meta-plans can refer to both domain plans or other meta-plans, as we shall see domain plans can only be accessed and manipulated via meta plans.

Except for the fact that they refer to other plans (i.e. they take other plans as arguments), meta-plan schemas are identical in structure to domain plan schemas. However, to allow these plans about plans, a vocabulary for referring to and describing plans will be needed. Developing a fully adequate formal model would be a large research effort in its own right. The development so far is meant to be suggestive of what is needed, and is specific enough for the preliminary implementation.

For example, to talk about the structure of plans a predicate PARAMETER (P, plan) will be assumed, which asserts that P is a parameter of the specified plan. A predicate STEP (action, plan) will also be used, to assert that the specified action is a step in the decomposition of the specified plan. The rest of the predicates will be introduced as they are needed.

Plans are not the only objects whose structure needs to be examined. In addition, there will be a need to refer to parameters of actions and propositions (for example, equality assertions) as well. Thus, the logic used will need to admit plans, actions, and propositions as objects. The PARAMETER predicate will be used to make assertions about the structure of all these types of objects.

The first two examples of meta-plans are given in Figure 2.3. INTRODUCE-PLAN takes a plan of the speaker that involves the hearer and presents it to the hearer, who is assumed to be cooperative. The way of introducing a plan given in the decomposition is to request one of the actions in the plan for which the hearer is the agent (the constraints). The definitions of speech acts such as REQUEST will be provided in the next section. Since the

hearer is cooperative, he or she will then adopt as a goal the joint plan containing the action (the first effect). The NEXT predicate (the second effect) informally means that the action so marked will be the next action to be executed in the plan. Similarly, a predicate LAST will be used to mark the action most recently executed. These predicates encode aspects of the discourse analysis and will be explained further in the next chapter.

HEADER:	INTRODUCE-PLAN(speaker, hearer, action, plan)
DECOMPOSITION:	REQUEST(speaker, hearer, action)
EFFECTS:	WANT(hearer, plan) NEXT(action, plan)
CONSTRAINTS:	STEP(action, plan) AGENT(action, hearer)
HEADER:	CONTINUE-PLAN(speaker, hearer, step, nextstep, plan)
PREREQUISITES:	LAST(step, plan) WANT(hearer, plan)
DECOMPOSITION:	REQUEST(speaker, hearer, nextstep)
EFFECT:	NEXT(nextstep, plan)
CONSTRAINTS:	STEP(step, plan) STEP(nextstep, plan) AFTER(step, nextstep) AGENT(nextstep, hearer) CANDO(hearer, nextstep, plan)

Figure 2.3: The Default Meta-Plans

Since this work is concerned with the development and use of meta-plans as a way of explaining conversational phenomena, it will be assumed that all meta-plans will be achieved via verbal communication with another agent. For example, other ways of introducing a plan, such as via a written contract, will be ignored throughout.

The second meta-plan, CONTINUE-PLAN, takes an already introduced plan defined by the WANT prerequisite and moves execution to the next step. One way this may be done is by asking (the REQUEST in the decomposition) the hearer to perform the next step, assuming

of course that the step is something the hearer actually can perform. This is captured by the decomposition together with the constraints. The effect will be that the portion of the plan to be executed is updated. This will be done with the predicates NEXT and LAST, which as mentioned above will also be useful for interacting with the discourse analysis.

As an example, consider an analysis of the following KL-ONE editor dialogue fragment using the above meta plans. (All such excerpts will come from the naturally occurring data unless otherwise noted.)

User:	Good morning. Please show the concept Person.
System:	Drawing...Ok.
User:	Add a role called hobby.
System:	Ok.
User:	Make the vr be Pastime.
System:	Alright
User:	Make a subc of Pastime called Sport...

Assume an edit plan involves accessing then performing a sequence of editing actions on a pre-existing concept. The first request of the user introduces a plan to edit the KL-ONE concept person. Each successive user utterance continues through the plan by requesting the system to perform the various editing actions. The first user utterance would thus correspond to INTRODUCE-PLAN (User, System, show the concept Person, edit plan). Since one effect of this INTRODUCE-PLAN is that the system adopts the plan, the system responds by executing the appropriate action in the plan, i.e. by showing the concept Person. The user's next utterance can then be recognized as CONTINUE-PLAN (User, System, show the concept Person, add hobby role to Person, edit plan), and so on for the other user utterances.

The above fragment is typical of the class of dialogues currently considered in plan-based approaches to language understanding. Systems are usually limited to discussion of one task,

with subdialogues corresponding only to execution of the subtasks. In terms of the proposed meta-plans, such dialogues would be modelled by introducing, then continuing through, a task plan. The two meta-plans thus make explicit some underlying mechanisms of the earlier works.

More importantly, these two meta-plans are part of a larger set of meta-plans enabling a uniform treatment of a wide range of subdialogues. As the data analysis illustrated, dialogues reflecting smooth plan execution are almost an exception rather than the norm. Instead, subdialogues often correspond to interruptions due to problems that arise during plan execution. A response that a speaker predicts will be easily understood might in actuality need clarification. Or, the system might perform an action based on incorrect assumptions, causing parts of the incorrectly executed plan to later be redone. Introductions, plan continuations, or even interruptions can themselves be interrupted. The remainder of the meta-plans were developed to formalize some of these ways in which the expected domain and meta plan execution is interrupted.

Figure 2.4 presents an example clarification meta-plan, IDENTIFY-PARAMETER, that helps identify a parameter that appears in another plan.

HEADER:	IDENTIFY-PARAMETER(speaker, hearer, parameter, action, plan)
DECOMPOSITION:	INFORMREF(speaker, hearer, term, proposition)
EFFECTS:	NEXT(action, plan) KNOW-PARAMETER(hearer, parameter, action, plan)
CONSTRAINTS:	PARAMETER(parameter, action) STEP(action, plan) PARAMETER(parameter, proposition) PARAMETER(term, proposition) WANT(hearer, plan)

Figure 2.4: A Clarification Meta-Plan

IDENTIFY-PARAMETER provides a suitable description of a parameter in the plan referred to that enables the hearer to execute an action in the decomposition of the plan. It is performed by describing the parameter via some description, using a proposition relating the parameter to the new description (INFORMEF will be further explained in the section on speech acts). It has several constraints on the relationship between the meta-plan and the plan it concerns, namely that *parameter* must be a parameter of an *action* that must be in the *plan*, and that the describing *proposition* will also involve the specification of *term*. Finally, the plan being clarified must already be a goal. The effect of this plan is the predicate KNOW-PARAMETER (agent, parameter, action, plan), defined to mean that *agent* has a description of *parameter* that is informative enough to allow *agent* to execute *action* in *plan*, all other things being equal. While the axiomatization of KNOW-PARAMETER is problematic, it shall only be used in simple cases where its use is straightforward.

For example, in the following dialogue fragment the first user utterance introduces a domain plan involving mounting tapes, i.e. INTRODUCE-PLAN (User, System, mount tape5, mount tapes).

User: Can I get tape5 on a drive?

System: Write enabled?

User: No. Also, while you're up, can I get tape4 without write enable?

However, since the system does not know whether the tape should be write enabled, the expected execution of the action is interrupted for initiation of a user clarification, i.e. INTRODUCE-PLAN (System, User, IDENTIFY-PARAMETER (User, System, read-only or write-enabled, mount tape5, mount tapes), clarification plan). The user responds with the IDENTIFY-PARAMETER, instantiating and executing its decomposition with INFORMREF (User, System, no write-enable, the mounting of tape5 is not write enabled), then resumes the interrupted plan to mount tapes. Note that rest of the user's second utterance now provides enough information to avoid a clarification.

Figure 2.5 presents the last class of meta-plans, those that debug plans that did not execute as expected.

HEADER:	CORRECT-PLAN(speaker, hearer, laststep, newstep, nextstep, plan)
PREREQUISITES:	WANT(hearer, plan) LAST(laststep, plan)
DECOMPOSITION-1: DECOMPOSITION-2:	achieve WANT(hearer, newstep) achieve WANT(hearer, nextstep)
EFFECTS:	STEP(newstep, plan) AFTER(laststep, newstep) AFTER(newstep, nextstep) NEXT(newstep, plan)
CONSTRAINTS:	STEP(laststep, plan) STEP(nextstep, plan) AFTER(laststep, nextstep) AGENT(newstep, hearer) ~CANDO(speaker, nextstep, plan) MODIFIES(newstep, laststep) ENABLES(newstep, nextstep)
HEADER:	MODIFY-PLAN(speaker,hearer, change, changee, newAction, oldAction, oldPlan, newPlan)
PREREQUISITE:	WANT(hearer, oldPlan)
DECOMPOSITION:	REQUEST(speaker, hearer, newAction)
EFFECTS:	POP(CLOSURE(oldPlan)) NEXT(newAction)
CONSTRAINTS:	PARAMETER(oldAction, changee) ~EQUAL(change, changee) STEP(oldAction, oldPlan) REPLACE(stack, oldStack) STEP(newAction, newPlan) EQUAL(newAction, SUBST(change, changee, oldAction)) EQUAL(TYPE(change), TYPE(changee))

Figure 2.5: Debugging Meta-Plans

CORRECT-PLAN inserts a repair step into a pre-existing plan that would otherwise fail. More specifically, CORRECT-PLAN takes a pre-existing plan having subparts that do not interface as expected during execution: the plan thus needs to be modified by adding a new goal to restore the expected interactions. The pre-existing plan has subparts *laststep* and

nextstep, where *laststep* was supposed to enable the performance of *nextstep*, but in reality did not. Thus the plan must be corrected by adding *newstep* to the executed plan, which enables the performance of *nextstep* and thus of the rest of *plan*. As in INTRODUCE-PLAN, the plan to be corrected can be introduced by a REQUEST for an as yet to be performed step (here, either *nextstep* or *newstep)*. The effects and constraints capture the plan situation described above and should be self-explanatory, with the following exceptions. MODIFIES (action2, action1) means that *action2* is a variant of *action1*, for example the same action with different parameters or a new action achieving the still required effects. ENABLES (action1, action2) means that the problematic preconditions of *action2* are in the effects of *action1*.

As in the last KLONE-ED fragment, in the following fragment the user is also executing an edit plan.

User: Good. Now put a part role on robot toes whose VR is unlabelled and which is superc'ed to physical objects, and under it put three generics labelled toe joints, nail catchers, and toe padding. That'll finish this little bit.

System: Drawing (sigh)...OK.

User: You forgot the cables.

In particular, the first utterance shown is a CONTINUE-PLAN (User, System, last User edit action, System put a part role on robot toes...., User edit plan). However, since the system executes the requested portion of the plan incorrectly (the goal of having the generics connected to the toe parts via cables was unmet), the user must interrupt execution of the editing task to correct the system. This is done via "You forgot the cables," eg. CORRECT-PLAN (User, System, System put a part role...., System add cables to result of previous put, next User edit step, User edit plan).

Finally, the last meta-plan to be discussed is MODIFY-PLAN, which replaces an incorrect plan with a modification of the plan. This is in contrast to CORRECT-PLAN, which just augments the original plan. More specifically, a new action is constructed from an incorrect action by replacing the filler of one of its parameters with a different value. A

modified plan is then constructed and re-executed by replacing the old action with its modification. These relationships are defined via the plan constraints. As in CORRECT-PLAN, the prerequisites indicate that the plan to be debugged must already be a goal. One way to perform this meta-plan is to request execution of the modified action. The POP effect and REPLACE constraint explicitly overrule the normal stack operations described in the next sections. Informally, instead of returning to the interrupted plan we instead re-execute a modification of this plan.

In the following fragment, the user's second utterance modifies the plan incorrectly executed on Wednesday (reintroduced by the user's first utterance).

User: On Wednesday, I created a rather lengthy listing on the line printer. I hope it hasn't been discarded. I neglected to ask anyone to hold it.

System: It has been discarded.

User: Okay. I'll do it again. Don't throw it away. I'll pick it up this afternoon.

More specifically, the user's second utterance is recognized as achieving MODIFY-PLAN (User, System, holding time of listing explicit, holding time of listing defaulted, today's create lengthy listing, Wednesday's create lengthy listing, plan using old listing, plan using new listing).

While many other ways of interrupting normal plan execution could be developed, recognition of the small set of meta-plans shown will be sufficient to understand several subdialogue classes in all three domains.

2.2. The Plan Stack

A *plan stack* will be used to monitor execution of a task plan and its various clarifications and corrections. During a dialogue, a stack of executing and suspended plans is built and maintained by the plan recognizer, each meta-plan referring to the plan below it, with the domain-dependent task plan at the bottom and the currently executing plan at the top. The

stack will thus encode the domain plan and various meta-plans introduced, their relationships to one another, and knowledge about which plans are currently executing and which will later be resumed. As will be seen in the next chapter, other models of discourse (e.g., Reichman [76], Polanyi and Scha [71]) have shown that topic structure follows a stack-like discipline. Within the plan stack, a single element corresponds to either a domain or meta-plan instantiation structure. In earlier systems, traversal of one such structure (which could itself be modeled with a stack (Grosz [37])) constituted the dialogue processing.

In this work, the stack of plans will always represent what the system believes is the state of the joint plan. Because both agents may construct and execute these plans, however, at times it will seem that the stack is not truly a stack. This occurs when the user acts and the system has to recognize what sequence of planning and execution steps the user did. For example, if the user popped the top plan, and executed a step in what is now the user's top plan, the system would recognize this as executing a plan in the second from the top plan. This anomaly is quickly resolved as the system can then pop its stack to bring the two agents' views back into synchronization. Thus, once the plan recognition process is completed, the observed action is always in the plan that is on the top of the stack.

To rephrase this, plans are added and deleted according to the stack discipline. The plan recognizer, however, is allowed to inspect the entire stack in order to recognize that the user has popped the stack before the user executed the recognized action. Even when the system believes the top plan has completed successfully, it cannot be popped before some acknowledgement from the user, thus allowing for a clarification of the complete plan. The acknowledgement could be explicit, but most often is implicit in that the user acts in such a way that the system recognizes that it must pop the top plan.

The stack can thus be viewed as having a mixture of both suspended (partially executed) and completed plans. At the top of the stack there is a set of plans (possibly null) that the

recognizer believes has been executed and completed. As just discussed, these plans cannot be eliminated as a possible topic until the user acknowledges their successful completion. Below these plans is at least one currently suspended plan. Each suspended plan will be resumed when the one above it is popped. In the case when there are no completed plans, the top suspended plan is also believed to be executing. Thus, the top of the stack is either an executing or just-executed plan. The rest of the stack may contain other suspended or completed plans. If the stack is empty, a task will either be introduced or has just been concluded. The actual pushing and popping of plans will be discussed in the section on plan recognition.

As an example, a clarification subdialogue is modeled by a meta-plan structure that refers to the plan that is the topic of the clarification. When a clarification plan is recognized, it is pushed onto the stack. The previous top, the plan being clarified, is temporarily suspended. When the clarification is complete and its success acknowledged, the stack is popped and resumption of the previous plan is recognized.

For example, consider the continuation of an earlier dialogue fragment, the fragment illustrating the INTRODUCE and CONTINUE meta-plans in the last section.

1) User:	Make a subc of Pastime called Sport.
2) System:	OK.
3) User:	Make an iconcept of Sport called Golf.
4) System:	OK.
5) User:	Is there a concept called mailing-address or something like that?
6) System:	There is no concept "mailing-address." What do you mean by "or something like that?"
7) User:	Anything that partially matches spelling of the concept name given or that identifies the concept indirectly via some other concept.
8) System:	There is a concept called Address. Is that of help?

Recall that at the point shown (utterances (1) and also (3)) the user is just continuing the previously introduced editing plan. With utterance (5) the user interrupts execution of this editing

to request a system clarification. This is necessary since to edit a concept the user has to be able to uniquely identify the concept. With the first part of utterance (6) the system begins the clarification, but then decides to suspend it to initiate another clarification subdialogue (regarding the first request for clarification), as done via the second part of utterance (6). In terms of the stack, the initial clarification was pushed on top of (and then suspended) the user's original edit plan. Similarly, the second clarification was then pushed onto the first. The state of affairs at this point is shown in Figure 2.6, where the top of the three element stack corresponds to the top of the page.

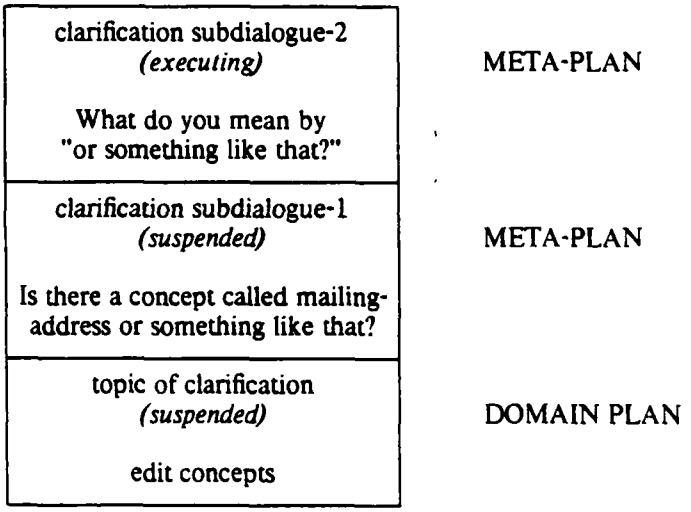


Figure 2.6: A Clarification Plan Stack

With utterance (7) the user completes the second (but executing) clarification subdialogue. The system can then pop the stack and complete the now resumed initial clarification subdialogue (via utterance (8)). Finally, the stack could then be popped once more and the domain plan resumed.

As mentioned above, in this work the stack will typically consist of a series of meta-plans, each referring to the plan below it, with a domain dependent task plan at the bottom. This

reflects the fact that in the data the majority of the interruptions refer to the previous topic, whether it be the original domain plan or one of its interruptions. However, when there are unrelated topics as in the following dialogue, a stack like the one in Figure 2.7 will be constructed.

User: Can I get tape5 on a drive?

System: Write enabled?

User: No. Also, while you're up, can I get tape4 without write enable?

System: Ok...

User: Is archive being run more than once per week, yet?

System: No.

System: Ok, tape5 is on drive1 and tape4 is on drive 2 without write enable.

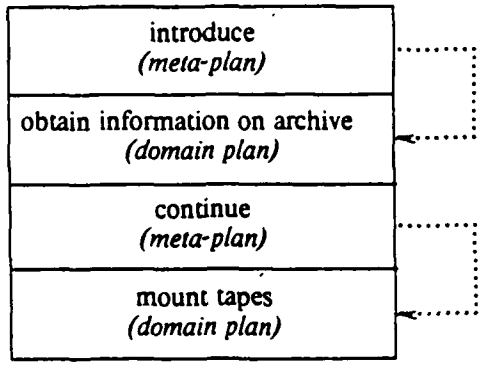


Figure 2.7: An Interrupting Topic Change Stack

In other words, the third line continues execution of a clarified domain plan, which as indicated by the fourth line will take some time. The user takes this opportunity to initiate discussion of an unrelated topic. This is the state of the stack shown. With the final system utterances the interruption is concluded (the top 2 plans popped) and execution of the tape mounting plan continued.

A stack metaphor obviously is an idealization for naturally-occurring conversations. For example, interrupted topics are not always returned to. Consider the following fragment (the continuation of Dialogue 2), where the user ignores the system's question:

User: Is there a role on employee called "retirement fund" or something like that?

System: No there isn't. What information are you trying to add?

User: How about a role called "pension program" or "pension plan?"

In terms of the plan recognition model, we shall see that while an interpretation that corresponds to the stack discipline is preferred to one that doesn't, if no such choice exists the non-stack-like behavior will be pursued. With respect to the above fragment, the system will eventually have to pop the incomplete clarification subdialogue in order to understand the user's last utterance.

As will be seen in the next chapter, the default stack mechanism can also be explicitly overruled by the particular phrasing of the dialogues. In cases like the above, phrases such as "never mind" could be used to signal non-resumption. Another type of divergence from the stack metaphor occurs with resumption of topics previously popped from the stack (and thus considered as completed topics). Linguistic devices exist to signal such unexpected behavior, for example prefacing an utterance with temporal phrases as in

Yesterday, you mentioned that the recipe calls for scallions, but you didn't list them as an ingredient today. [50]

Thus, the default mode of plan recognition will prefer stack-like interpretations whenever it has a choice, unless the surface linguistic phenomena explicitly indicate otherwise. However, one type of non-stack-like behavior that the model will not be able to handle is illustrated by the following fragment.

L: Hi M. Let's cook.

M: Right! And with gas! OK, here is a list of ingredients:
1. Eggplant

2. Garlic
3. A small piece of ginger
4. Some hot peppers
5. Hoi sin sauce

Ok, any questions?

L: Hi, yes, there's one small thing that might cause trouble. We have ginger powder, not real ginger.

M: I don't think they can really be substituted because the ginger is supposed to lend a subtle flavor to the oil, but you can do without OK, no problem...

L: It's not a question of doing totally without. We have ginger powder.

M: The first thing to do is to peel the eggplant into about 8 lengthwise pieces and then halve them.

L: OK, will do. Be back soon.

M: Right, but I sorta think that ginger powder isn't the same as sliced ginger.

L: It's obviously not Liebnizian identical to it, but I think it will do.

M: Let me know when you have finished with the eggplant. By the way, the idea is to make the eggplant slices into bite size morsels. So use your judgement really.

L: Thanks for the tip. We were just wondering about that.

M: If you want to use ginger powder instead of real ginger, go right on ahead! I'll be interested in the results.

L: The eggplant has been sliced...

In this fragment the discussion of the ginger powder versus the ginger pieces is interleaved with the execution of the steps in the recipe. This could not be explained via a stack mechanism except by continuously popping a topic (viewing it as finished) and then immediately pushing (or reviving) it, which does not really capture the sense of multiple active topics. While such dialogues are possible, they are fairly uncommon in the data. To many people they also seem "ill-formed."

3. Speech Acts

Speech act theory (Austin [9], Searle [84], Grice [35]) views utterances as actions that achieve intended effects, rather than as statements that are either true or false. For example, Austin noted that utterances such as "I christen this ship the Queen Nancy" actually change

the state of the world, rather than just assert something that is true, as in "Grass is green." Searle expanded on this idea, categorizing the various types of speech acts and analyzing the conditions under which they may be successfully performed. Allen, Cohen and Perrault [3, 22] used a language planning system to computationally formalize some of these ideas. This section will present the definitions of the speech acts used for this work, based on the definitions given in Allen and Perrault [3].

HEADER:	REQUEST(speaker, hearer, action)
PREREQUISITE:	WANT(speaker, action)
DECOMPOSITION-1:	SURFACE-REQUEST (speaker, hearer, action)
DECOMPOSITION-2	SURFACE-REQUEST(speaker, hearer, INFORMIF(hearer, speaker, CANDO(hearer, action)))
DECOMPOSITION-3:	SURFACE-INFORM(speaker, hearer, ~(CANDO(speaker, action)))
DECOMPOSITION-4:	SURFACE-INFORM(speaker, hearer, WANT(speaker, action))
EFFECTS:	WANT(hearer, action) KNOW(hearer, WANT(speaker, action))
CONSTRAINT:	AGENT(action, hearer)
HEADER:	INFORM(speaker, hearer, proposition)
PREREQUISITE:	KNOW(speaker, propostion)
DECOMPOSITION:	SURFACE-INFORM(speaker, hearer, proposition)
EFFECTS:	KNOW(hearer, proposition) KNOW(hearer, KNOW(speaker, proposition))
HEADER:	INFORMREF(speaker, hearer, term, proposition)
PREREQUISITE:	KNOWREF(speaker, term, proposition)
DECOMPOSITION:	achieve KNOW(hearer, proposition)
EFFECT:	KNOWREF(hearer, term, proposition)
CONSTRAINT:	PARAMETER(term, proposition)
HEADER:	INFORMIF(speaker, hearer, proposition)
PREREQUISITE:	KNOWIF(speaker, proposition)
DECOMPOSITION-1:	achieve KNOW(hearer, proposition)
DECOMPOSITION-2:	achieve KNOW(hearer, ~proposition)
EFFECT:	KNOWIF(hearer, proposition)

Figure 2.8: Speech Act Definitions

As shown in Figure 2.8, speech acts are formalized as plan schemas, using the notation developed in previous sections. For example, the first speech act is a request from the speaker to the hearer for an action. The constraint specifies that the hearer is the agent of the action.[2]

[2]Technically this is only true for the first two decompositions. When the third decomposition (and sometimes the fourth) is used, the agent changes from the speaker in the decomposition to the hearer in the header. For example,

The decompositions indicate several typical *surface linguistic acts*, templates for actual utterances that could be used to execute the speech act. For example, if the speaker wanted the hearer to mount a magtape, any of the four following utterances could be used to convey the request:

"Mount a magtape."	(decomposition 1)
"Can you mount a magtape?"	(decomposition 2)
"I can't mount a magtape."	(decomposition 3)
"I want to mount a magtape."	(decomposition 4)

Note, however, that only the first decomposition literally conveys the intended request. Unlike Allen and Perrault [3], decompositions have been modified to include the conventionalized forms of *indirect speech acts*, speech acts that are realized through surface forms that literally appear to mean something else. Although such inferences could be derived from first principles (Allen and Perrault [3]), those issues will not be addressed here. Finally, the first effect of REQUEST is based on the assumption that the hearer is cooperative (see Cohen and Perrault [22] for a formulation where this assumption is not made). The second effect is new, and explicitly asserts that the hearer then believes the preconditions held if the act is done successfully. This could again be inferred from first principles, but adding it to the definition allows the use of a simple plan recognition algorithm throughout.

The treatment of INFORM and its two other variants is similar. The typical INFORM speech act is a declarative sentence, where the speaker tells the hearer something that the speaker but not the hearer knows. An example is "The train leaves at eight-fifty." INFORMREF and INFORMIF are two variations needed to handle wh-questions and yes/no questions, respectively. For example, "When does the train leave?" is a REQUEST to INFORMREF, and "Does the train leave at 8:50?" is a REQUEST to INFORMIF. The only difference from Allen and Perrault [3] is that there is an extra parameter to INFORMREF and KNOWREF. The assertion KNOWREF (agent, term, proposition) means that *agent* knows a

description of *term*, which satisfies *proposition*.

This is simply a notational variant that is closer to the actual implementation. Thus, rather than stating the goal to know when train TR1 leaves as

KNOWREF (agent, the x: depart-time (TR1, x))

as in Allen and Perrault [3], we write

KNOWREF (agent, ?time, EQUAL (depart-time (TR1), ?time)),

where "?time" is a variable of type time.

Not all such assertions involve the equality predicate. For example, the representation of the goal behind the utterance "What do you want?" would be

KNOWREF (speaker, ?action, WANT (hearer, ?action)).

This operator can be formally defined within a possible worlds semantics of the BELIEF operator by using "quantifying in" as done in Allen and Perrault [3]. While this analysis is not fully satisfactory, it is adequate for the present purposes.

As in Allen and Perrault [3], determination of the literal surface linguistic act is fairly straightforward. The surface speech act is correlated with sentence mood as well as particular words. Imperatives indicate SURFACE-REQUESTS, declaratives SURFACE-INFORMS, and interrogatives SURFACE-REQUESTS to INFORM. Words such as "when" can further restrict questions to a SURFACE-REQUEST to INFORM of a time. The propositional content of the surface acts (e.g. *action* in SURFACE-REQUEST and *proposition* in SURFACE-INFORM) can be determined via the standard syntactic and semantic analysis of most parsers.

A new surface form called SURFACE-NP has also been included. This allows a simple treatment of sentence fragments such as definite noun phrases. As with indirect speech acts, determination of the underlying speech act is left to the plan recognizer. For example, a

---

this happens when "I can't reach that book" requests the tall hearer to reach it instead.

SURFACE-NP is yet another way of executing a REQUEST. More formally, the REQUEST (speaker, hearer, action) schema would be as above, with a new decomposition of SURFACE-NP(speaker,hearer,noun-phrase) and an added constraint CONTAINS (action, noun-phrase), where CONTAINS states that the action involves the noun phrase as a parameter or recursively, as a parameter of a parameter. An utterance such as "Track eleven?" would be parsed as a SURFACE-NP (passenger, clerk, track11), which could be recognized as a REQUEST(passenger, clerk, action (...track11...)). Since the utterance was a question, besides indicating a REQUEST *action* can be further constrained to be an inform. Carberry [17] has independently proposed a similar idea for the representation of fragments. For example, her semantic representation of "Track eleven" would be the proposition genpred(Track11), which "indicates that the name of the specific plan proposition is as yet unknown but that one of its parameters must associate with the constant" track eleven.

As mentioned above, it will be assumed that all meta-plans are done using speech acts. For example, another way to achieve KNOWREF goals would have been to look up the answer in a reference source. At the train station, for example, one can find departure times and locations from a schedule.

4. Plan Recognition

Plan recognition is the process of inferring an agent's plans and goals from utterances or physical actions, the effects and methods of achieving such goals. Such a process involves not only recognizing an initial plan, but also deciding whether subsequent utterances are related to the same plan as opposed to a new one.

Assumptions often made in plan recognition depend on certain relationships between the observer and the agent being observed. In *intended plan recognition*, the agent being observed not only knows of the observation, but also performs actions that are intended to facilitate the observer's recognition process. The speech act view of communication (Grice [35], Searle [84])

and systems based on it (Allen and Perrault [3] , Sidner and Israel [86]) are examples. Each agent is aware of what inferences the other could make, given their mutually believed plan libraries. Thus, the speaker constructs utterances enabling the desired inferences. The hearer in turn views the inferences made as intended to be made, and thus acts on what the speaker intends for the hearer to think about the hearer's wants. This is in contrast to *keyhole recognition* (Cohen et al [23]). Here the agent being observed is not aware of the observation and thus does not structure behavior in a way that facilitates plan recognition, as when an observer watches an agent through a keyhole. The BELIEVER system [83] and PAM [96] infer in this manner. Helpful behavior provides unintended (as well as intended) responses and thus relies on both types of recognition. For example, given a question such as "When does the train to Montreal leave?", "Eight-fifty. Gate seven" would be a helpful response since it provides more information than requested. As will be seen below, assuming an intended mode of plan recognition will help justify making it an incremental process.

In this work, the plan recognizer attempts to recognize the meta-plan, and thus the object domain or meta-plan, that led to the production of the input utterance. The plan recognizer has at its disposal a library of domain and meta-plan schemas, the representation of the parse of the input utterance, and the plan stack representing the current state of the dialogue. Using the plan recognition algorithm, the recognizer will then output a modified plan stack, representing as much of the updated plan state underlying the dialogue as can be unambiguously determined. An utterance either continues an existing plan on the stack or introduces a meta-plan to some plan on the stack. If either of these is not possible for some reason, the recognizer hypothesizes a plausible plan using any of the plan schemas. At the beginning of a dialogue there is no stack, so the general expectations from the task domain are used to guide the plan recognizer. For example, a train clerk expects questions about boarding and meeting trains. The plan recognizer performs its task using an incremental heuristic search.

4.1. Forward Chaining

The plan recognizer's task is to find a sequence of instantiations of plan schemas, each one containing the previous one in its decomposition,[3] that connects every utterance to an expected meta-goal. More specifically, the system tries to find plans in which the utterance is a step, and then tries to find more abstract plans for which the postulated plan is a step, and so on. Since every meta-plan takes other plans as arguments, recognition of any meta-plan will need a recursive recognition on the argument plan introduced. For example, suppose a speaker asked to buy a train ticket. A search through the decomposition of the meta-plan schemas indicates that this request may be a *way of* introducing a plan to buy a ticket. Chaining from introducing a plan does not yield any higher level goals. The same search process is then performed on the introduced plan to buy a ticket. Searching through the plan library shows that this act could be a step in a plan to take a train trip, which is itself not a step in any other plans in the library. Since taking a trip is a domain plan, no other plans are introduced and recursive chaining halts.

Once a set of plans is recognized, each is expanded top-down by adding the definitions of all steps and substeps until there is no unique expansion for any of the remaining substeps. Each plan is then pushed onto a stack so that the original meta-plan is on top, every meta-plan refers to the plan below it, and the domain dependent plan is on the bottom.

While this search is a simple tree climbing process and thus theoretically terminates, if unconstrained it can be both costly as well as unable to yield a unique plan interpretation. The search process needs to be controlled with various heuristics, as well as limited by dividing it into incremental stages.

---

[3]Plan chaining can also be done via effects and preconditions. (Pollack [73] is even extending these types of links to enable recognition of non-existent library plans). To keep the examples simple, all plan schemas have been expressed so that chaining via decompositions is sufficient.

4.2. Heuristics

4.2.1. Coherence Heuristics

The search process described above is too general, for it does not take into account the influence of the portions of the previous discourse context still being discussed or interrupted (as maintained on the stack). The following ordered heuristics control the search process by preferring sequences that correspond to the most coherent continuations of the dialogue. If during the search process the observed action can be incorporated into a plan according to one of the following three ordered preferences, the chaining stops:

(1) by a continuation of the executing plan on the stack (i.e. recognition of CONTINUE-PLAN)
(2) by introducing a clarification or correction meta-plan to any plan on the stack
(3) by constructing meta-plans and associated object plans that are plausible given the domain-specific expectations about plausible goals of the speaker

Thus the recognizer prefers an utterance interpretation that continues a plan rather than suspending one for its clarification or correction, which is more coherent than introducing a new plan altogether.

Preference (1) involves situations where the agent does exactly what was expected in the given situation. The most common example of this occurs in answering a question, where the answer is explicitly expected. As another example, if the agent was observed going to the ticket window and paying for a ticket, the BUY plan would be postulated. If the agent is next observed receiving the ticket, it would be recognized as a continuation of that BUY plan. A more complex example is illustrated in Figure 2.9, where a BOARD subplan connects an observed GOTO action with an expected (stacked) TAKE-TRAIN-TRIP goal. (The notation represents the hierarchical structure of a plan instantiation as a tree).

While preference (1) involves expanding and executing the plan on top of the stack, preference (2) allows for suspension of the top plan. For example, an agent may require more

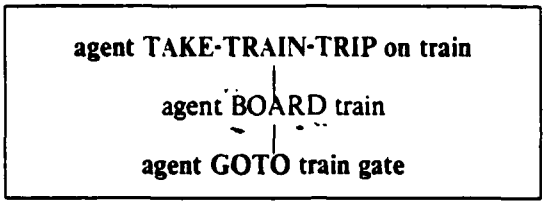


Figure 2.9: Introduce a Subplan to Continue the Executing Plan on the Stack

information in order to actually execute the next step, and will thus need to temporarily suspend execution while engaging in a clarification. Of course, this clarification may itself be suspended by another clarification during execution, and so on. Preference (2) thus involves not only recognizing a clarification meta-plan based on the utterance, but also, in satisfying its constraints, connecting the meta-plan to a plan on the stack. If the plan on the stack is not the top plan the stack must be popped down to this plan before the new meta-plan is added to the stack. If the plan that is the object of the clarification is ambiguous, the alternative closest to the top of the stack is preferred. For example, if a BUY plan is on the top of the stack, the utterance "How much does the ticket cost?" could be recognized as a request for a clarification of the PAY subplan. The clarification plan would be placed on the stack and the BUY plan suspended as shown in Figure 2.10. Recall that the stack represents joint plans and may involve multiple agents. The recognizer not only reconstructs the goals of the hearer, but by being cooperative automatically adopts them as its own goals as well.

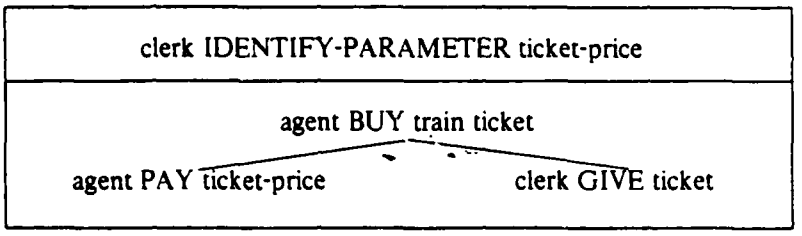


Figure 2.10: Introduce a Clarification Meta-Plan to a Plan on the Stack

Preference (3) may involve not only introducing a new plan but also, if it is a meta-plan, using the constraints to recursively introduce a plausible plan for the meta-plan to be about. This occurs most frequently at the start of a dialogue or topic shift, i.e. when the previous plan context either does not exist or is ignored. Suppose a speaker begins a dialogue with "I want to buy a ticket to Montreal." The utterance is recognized as an explicit plan introduction, a meta-plan with constraints that enable the recognition of the plan being introduced as well. Figure 2.11 shows the stack constructed out of these plans. As desired, the plans are placed on the stack in the order they were generated rather than the order they were recognized.

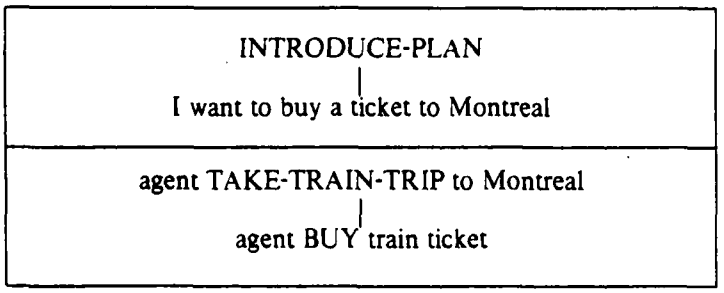


Figure 2.11: Recognizing Multiple Plans from One Utterance

Note that each preference involves not only recognizing a meta-plan based on the utterance, but in satisfying its constraints, also involves connecting the meta-plan to an expected plan (which is either an already stacked plan or an introduced plausible domain plan).

While these heuristics and their ordering have not been validated with psychological experimentation, they have intuitive appeal. For example, if the first heuristic was always applicable, the discourse behavior would default to the earlier systems in which a dialogue contains no interruptions. Preferring the second heuristic over the third corresponds to the view that without any explicit linguistic markings (as discussed in the next chapter), topic change is always least expected. Thus, while interruptions are not generally predicted, they can be handled when they do occur. These heuristics also follow the principle of Occam's razor, since they are ordered as to introduce as few new plans as possible. Other models of discourse (e.g., Carberry [15], McKeown [64]) use similar heuristics.

4.2.2. Plan Based Heuristics

Candidate plans are also eliminated by a set of heuristics based on assumptions regarding rational planning behavior. These heuristics are applied to each candidate plan after every step of the chaining process. For example, as in Allen and Perrault [3] plans that are postulated whose effects are already true are eliminated, since achieving these plans would produce no change in the state of the world. Similarly, it would be foolish to postulate plans containing preconditions that could not have been met or constraints that cannot be satisfied. Thus, if a clerk in an information booth was asked "The eight-fifty to Montreal?" a likely response would be "Gate 7" but not "Ten dollars." However, "Ten dollars" would be a likely response if the clerk in a ticket window heard the same question. This is because the constraint of the buy-ticket plan states that the hearer must be able to sell tickets. Thus the information clerk thinks that the speaker is boarding a train rather than buying a ticket. Applicability of these heuristics, of course, depends on our assumption that the speaker and hearer share the same knowledge regarding plan schemas.

Allen and Perrault also have heuristics preferring candidate plans closest to the expectations. The coherence heuristics discussed above greatly elaborate on this idea. Within these

coherence preferences there also exists a heuristic that eliminates plans if setting unknown constants in the plan being recognized equal to known constants in the plan previously recognized causes a contradiction. Thus, in "I'd like to go to Montreal. Where is the gate?", only plans that postulate that the requested gate is equal to the gate of the train to Montreal are not eliminated.

4.3. Incremental Search

Even after application of all heuristics (and as we will see in the next chapter, linguistic constraints), the search may still lead to several possible plans at the same preference level and thus be potentially explosive. The strategy chosen here is that after reaching such branch points, the search is terminated until input of the next utterance. For example, since both BOARD and MEET plans can be recognized from a GOTO (recall Figure 2.1), chaining would halt with two branches if both plans were plausible (even though further chaining is possible). Such premature termination is typical in dialogue processing (Sidner and Israel [86], Carberry [15]), since later utterances in the dialogue often eliminate many of the branches. This will be seen in one of the examples in Chapter 4. This is in contrast to question-answering systems such as Allen and Perrault [3]. Since the input is all that exists and the system needs to provide an immediate response, they do not have the luxury of waiting for further input and either cannot or prefer not to initiate a clarification subdialogue. Instead such systems often use fairly ad-hoc probabilistic ratings to rank the options and control the search. In cases of intended plan recognition, Cohen, Perrault and Allen [23] argue that incremental recognition is also philosophically as well as pragmatically justified.

> A special heuristic is useful to control *intended* plan-recognition inferences. It is based on the assumption that the speaker is a rational agent, and thus only intends inferences to be drawn if they can be drawn unambiguously. The heuristic therefore terminates intended inference chains that lead to mutually exclusive alternatives, for which the hearer has no reason to select one over the others. Of course, the success of this heuristic depends on the accuracy of the models the speaker and hearer maintain of each other, a not unreasonable condition. [23]

Since it is assumed that the hearer is provided with appropriate information to recognize what

is intended, any decisions beyond branch points are indeed arbitrary. Note, however, that incremental recognition will only be used for plan branch points and not for speech act recognition. For example, in the current formulation determination of an underlying speech act appears to be a side effect of the plan recognition process.

Thus, the bottom-up inference process halts when we have either incorporated the utterance into our expectation structure (satisfied a coherence heuristic) or the search space becomes too large. In the latter case, multiple stacks are created to record each branch and the plan remains ambiguous. There are then several ways one might be chosen over the others. If it is the hearer's turn in the dialogue (i.e., no additional utterance is expected from the speaker), then the hearer may initiate a clarification subdialogue. If it is still the speaker's turn, the hearer may wait for further dialogue to distinguish between the possibilities.

5. Summary

A plan recognition system based on a library of domain specific plan schemas, a library of domain-independent meta-plan schemas, and a context-dependent, incremental recognition algorithm has been presented. The algorithm is applied after each user utterance, and constructs or updates a stack representing the joint plans of the user and system. More specifically, an initial stack is constructed containing a domain plan to be executed. Such a plan is expanded and executed until suspended for a new plan pushed onto the stack. This new plan is itself subject to suspension. When an interrupting plan is concluded, it is popped from the stack and the suspended plan below it resumed. Only as much of the stack as can be unambiguously determined is constructed during any given application of the algorithm.

The recognition algorithm is simple and general, yet since it is highly constrained it is also tractable. In choosing a best explanation for an observation, constraints arise from several sources - coherence with the previous dialogue, simplicity of the structures postulated, and conformity with principles of rational planning behavior. Furthermore, many complications of

earlier systems are eliminated by declaratively encoding knowledge of the plan execution process in meta-plans.

The formalization and use of the meta-plans also allows a clean, well-specified mechanism for plan execution issues, particularly plan suspension and resumption. The plan recognizer is unique in explicitly computing not only the goal being pursued, but also the execution relationship with the previous plan structures on the stack. This latter information is captured in the meta-plan, and connected to the former via constraint propagation.

Chapter 3

Discourse Analysis

1. Background

Researchers in artificial intelligence, as well as in various branches of linguistics and the social sciences, often study an utterance in the context of the discourse within which it occurs. Such an analysis has proven useful for improving explanations underlying *surface linguistic phenomena*, the actual lexical items and syntactic structures used in an utterance. It appears that such phenomena are correlated with particular discourse functions, that is, many phenomena regarded as syntactic are controlled by non-syntactic factors as well. For example, mode of reference is related to an object's changing status during a discourse. Early discourse work in artificial intelligence was done by Grosz [37], who used the influence of a global, hierarchical discourse structure to explain referring expressions in task-oriented dialogues. Linde [55] had similar results using apartment descriptions. Grosz and later Sidner [88] and Grosz, Joshi, and Weinstein [39] were also concerned with surface linguistic phenomena that appeared to be influenced by a more local discourse context. Reichman [76] developed an alternative model that was not limited to any particular dialogue genre and also explained a wide variety of referring expressions. Besides mode of reference, other lexical items proved to serve the important discourse function of marking shifts in or interruptions of units in a discourse structure. Grosz [37], Reichman [75, 76], Cohen [25], Polanyi and Scha [71], and

Grosz and Sidner [40] have investigated these *clue words* or *discourse markers* for a wide variety of genres. For example, a phrase such as "by the way" often precedes interruptions of a topic.

Discourse context has also proven useful when understanding utterances with missing words or phrases, for example sentence fragments and other elliptical utterances. While such inputs appear ill-formed at the sentence level, often they can be understood by using the preceding dialogue to fill in the missing parts. The most common approach is to view the problematic utterance as a reformulation of part of an old utterance. The interpretation of the elliptical utterance is then the interpretation of the old utterance, modified with the appropriate replacement. The Lifer system [44], Planes [94], Weischedel and Sondheimer [95], and Grosz [37] take this approach, using both syntactic and semantic information as the basis for the substitution. Carbonell and Hayes [19] extend these results by using a case-frame analysis. For example, case frames could handle utterances with more than one semantically similar fragment, and utterances that did not preserve linear ordering. Another approach is to use expectations based on not only language, but also world knowledge. For example, Granger [34] uses situational and intentional expectations to fill in unknown conceptual cases of verbs. Sometimes, however, the missing portion can not be recovered from the previous dialogue at all. A special case of this occurs when a dialogue begins with a sentence fragment, e.g. "The eight-fifty to Montreal?" Allen and Perrault [3] have shown how knowledge of a speaker's underlying intentions can be used to understand such initial sentence fragments. Carberry [17] shows how such planning knowledge, combined with knowledge of discourse goals, can also be used to process intersentential ellipsis.

Another area of concern has been the development of a set of domain-independent, linguistic relationships that relate units in a discourse. Sets of such relationship have been called *conversational moves* (Reichman [76]), *rhetorical predicates* (McKeown [64], Mann [60], Woolf and McDonald [99]), or *coherence relations* (Hobbs [46]), and have been developed from

examinations of naturally occurring texts. For example, in a well structured argument a claim might best be followed by a *support* for the claim. To understand an argument this implicit support relationship must also be understood. If achieving other discourse goals, the same information might be structured differently. Furthermore, limiting the ways such relationships can interact (for example by using schemas or grammatical productions) provides a useful mechanism for understanding or generating texts that are rhetorically well organized. Unfortunately, such mechanisms cannot handle naturally-occurring texts that aren't structured so nicely. Another problem with this approach is that in general the semantics of the predicates are very subjective, making it hard to algorithmically correlate them with specific surface utterances.

While the work described above could add another dimension to plan-based analyses of discourse, such a synthesis has been rare. Grosz [38], Levy [54], and Appelt [8] (inspired by Halliday [42]) suggested that actions achieve multiple goals, for example, intentional, discourse, and social (e.g. politeness) goals. While the system of Appelt formalized some of the multiple perspectives advocated by Grosz and Levy, discourse knowledge was only used to facilitate reference, generally in a single utterance. Early work on the ARGOT system [4] also attempted to integrate discourse and intentional goals. Sidner [90] argues that a plan recognizer able to recognize relationships between plans must use discourse markers; however, her system does not actually show to to coordinate this. Most recently, Grosz and Sidner [40] propose that to properly explain interruptions a framework incorporating intentional, attentional, and linguistic structures is necessary. Unfortunately, many of the particular details of such a theory have not yet been developed.

In this chapter we will see one way of merging discourse analysis and a plan recognition framework. As above, surface linguistic phenomena will provide clues regarding the underlying structure, here an intentional structure. The plan recognizer will use such information to provide further constraints on its processing. In contrast, most of the complex reasoning about

implicit relationships between utterances[1] will be totally re-expressed in terms of meta-plans and the plan recognition process. This will provide a formalization and tractability generally not available in the above works. Since this work concentrates on incorporating the domain-independent linguistic results into a plan recognition framework, rather than on a new investigation of the linguistic issues, each section will first elaborate on the relevant linguistic results and then explain how they either interface with or are incorporated into a theory of plan recognition.

2. Focus of Attention

Grosz [37] noted that in task-oriented dialogues, the task structure could be used to guide the discourse structure. She developed the notion of *global focus of attention* to represent the influence of the discourse structure. Without such selective consideration the knowledge necessary for understanding in even simple domains becomes overwhelming. For example, focus proved useful for limiting the search space during the resolution of definite noun phrases, as well as providing an alternative to recency explanations of pronoun resolution (Winograd [98]). The following excerpt (Grosz [37]) illustrates the usefulness of a hierarchical dialogue segmentation (corresponding to the task hierarchy).

Expert: Good morning. I would like for you to reassemble the *compressor.*
...
Expert: I suggest you begin by attaching the pump to the platform.
... (other subtasks)
Expert: Good. All that remains then is to attach the belt housing cover to the belt housing frame.
...
Apprentice: All right the belt housing cover is on and tightened down.
(30 minutes and 60 utterances after beginning)
Expert: Fine. Now let's see if *it* works.

The pieces of dialogue between the italicized pronoun and its referent correspond to subtasks, which although recent are not salient and thus not relevant for pronoun resolution. A

[1] An utterance will correspond to a single unit in the discourse structure (e.g. to a meta-plan). The analysis of specific dialogues, for example the correction dialogue in the next chapter, will show how multi-sentential utterances making a single point (e.g. "Can you mount a magtape? It's tape 1." as opposed to "Can you mount tape1?") are analyzed.

computational representation of this highlighting was achieved by segmenting the knowledge base into hierarchically structured focus spaces, corresponding to the discourse structure at a given point in the dialogue. The focus spaces contained the items explicitly mentioned in the subdialogue as well as items necessary for understanding and thus implicitly present, i.e. they contained all the salient items at a particular point in a dialogue. Moreover, Grosz observed that task-oriented dialogues subdivided into units just as the tasks subdivided into subtasks. Thus, such a representation could be computationally updated; the task structure could be used as a guide for discourse structure shifts.

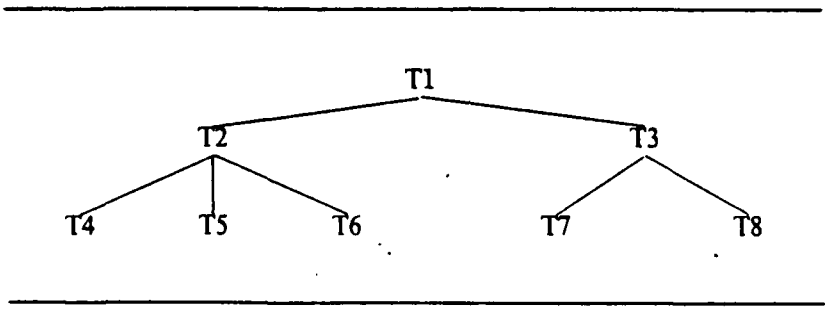


Figure 3.1: Task Structure and Discourse Structure

For example, Figure 3.1 illustrates how the task structure predicts dialogue *pops*, returns to discussions of higher-level tasks.

> When task T6 is completed, there is a return to the focus of T2 and possibly directly to T1. Objects that participate only in T4 or T5 are not in focus. Similarly, objects in T2 or T4-T6 cannot be directly referenced from T7 or T8. When T8 is completed, there may be a "pop" up to T3 or T1. (Grosz [37])

Global focus needs to be distinguished from *immediate focus* (Grosz [37], Sidner [88]) or *centering* (Grosz et al [39]), which represents the influence of the linguistic form of the utterance.

The idea of global focus, i.e., that at any point in a dialogue some items are more salient than others, has been incorporated with some modification into my plan recognition system.

Each plan on the stack has a marked subplan that is under execution or will be executing when the plan is resumed. This subplan is indicated using predicates as follows:

NEXT (subplan, plan) -- true only if *subplan* is the part of *plan* that is currently being executed, or will be executed when the plan is resumed;

LAST (subplan, plan) -- true if *subplan* is the last (i.e., most recent) part of *plan* to be executed.

These subplans will often be referred to as the part of the plan that is *in focus*, drawing an analogy with the discourse models above. In other words, when a plan is recognized the step that was executed is marked as in focus. When a plan is in focus, so are the items explicitly and implicitly involved in the plan. However, unlike in Grosz, the plan that is recognized from an utterance is always a meta-plan and thus introduces an associated object plan. Among the effects of every meta-plan are assertions that update these predicates with respect to the plan referred to. Recall INTRODUCE-PLAN, as defined in Figure 2.3. When recognized from a request, the plan recognizer will mark the executed request as in focus, using LAST. Furthermore, the particular action (subplan) introduced in the object plan will also be marked as in focus, via the predicate NEXT, as an effect of the plan.

As Grosz noted, the items in focus switches during the course of a dialogue, corresponding to shifts in the subplans currently being executed. Typically, hierarchical plans are represented as trees and executed depth-first as discussed with respect to Figure 3.1 above. We can now see that the CONTINUE-PLAN meta-plan declaratively encodes this control strategy. In particular, preference (1) of the coherence heuristics corresponds to simply following the task structure in the top plan. The other two preferences extend the shifting of focus in ways that allow interruptions, as will be described below. By making this knowledge explicit, both subtask and non-subtask (e.g. clarification, correction, and topic change) subdialogues can be treated uniformly.

Thus, the idea of global focusing proves useful at a purely intentional level by making plan recognition context dependent (and more efficient since the search space is constrained). The notion of focus is used solely at this level in Carberry [15], Sidner and Israel [86], and Carver et al. [20]. Carberry [15] provided explicit plan recognition rules for tracking shifts in the task structure. From an utterance, she recognized part of the task plan, which was then used to provide expectations for future plan recognition. For example, upon completion of a subtask, execution of the next subtask was the most salient expectation. However, the utility of the idea of focus was also due to its connection with surface linguistic phenomena. How this can be done in a plan recognition system will be discussed in the next section.

Finally, as mentioned above, global focus needs to be distinguished from immediate focus, which is also useful for explaining surface phenomena such as pronouns. In this work an immediate focus system will be assumed, the output of which is input to the plan recognizer along with the representation of the parse. The clarification subdialogue example in the next chapter will illustrate the details of this interaction, as well as what would happen if an immediate focus system were not available. In this latter case we will see that while the plan recognition task is harder, the pronoun resolution will ultimately be made as the plan is connected with its context.

3. Surface Linguistic Phenomena

Grosz [37] noted that shifts in the discourse structure were not only marked by shifts in the task structure, but also by various surface linguistic phenomena. As discussed above, global focus was useful for limiting the search space during the resolution of definite noun phrases. However, when a resolution could not be made using the items currently in focus, the definite noun phrase explicitly marked a shift in the focus. Other non-intentional clues to shifting were explicit words like "ok," which marked a discourse pop.

Reichman [76] was concerned with explaining linguistic phenomena totally non-intentionally. She developed a model that was not limited to task-oriented dialogues, and accounted for a much wider range of discourse popping (e.g., topic switch, a popping between plans rather than solely within plans). In her theory, she also noted that the non-linear structure underlying a dialogue was reflected by the use of surface phenomena such as referring expressions. The type of expression (e.g., pronoun, definite noun phrase, etc.) used to refer to an object was shown to reflect the object's degree of focus. She also showed how *clue words* provided clues to changes in the underlying discourse structure. Clue words signaled a boundary shift between the discourse units hierarchically structured as well as the kind of shift. For example, the clue word "now" indicated the start of a new unit that further developed the currently active unit, "by the way" indicated an interruption of the currently active unit for a totally new unit, while "anyway" indicated a pop of the interruption for a return to a previously active unit. (Note how this popping contrasts with the popping discussed in Figure 3.1. It is critical to keep the stack implicit in a depth-first traversal of a plan separate from the explicit stack of such plans). Reichman's model thus illustrated that spontaneous dialogues were indeed highly rule-governed rather than unstructured. In particular, rules of effective communication governed the use of clue words and choice of reference.

Many other researchers have also shown the usefulness of such clue words. In argument understanding, Cohen [25] claims clue words reduce the processing load of the hearer and are also necessary to overrule a small set of default argument structures. Polanyi and Scha [71] show how syntactic as well as other clues (e.g. change in gaze or intonation) are useful for their theory of discourse, particularly for allowing interruptions. Sidner [90] and Grosz and Sidner [40] claim that many interruptions and returns to interrupted topics cannot be understood unless they are explicitly marked as such.

The idea that surface linguistic phenomena in the input utterance can be used to help recognize and track changes in the underlying structures of the theory will be crucial to this

work as well. With respect to definite noun phrases, if the plan recognizer cannot match the input with objects in the focused plans, the other preferences will be applied and thus a new plan found, explicitly shifting the plan in focus.

With respect to clue words, correlations between specific words and the meta-plans discussed earlier will be available to the system. For example, the following is a list of a few clue words and the meta-plans they mark:

"how about"	INTRODUCE-PLAN, MODIFY-PLAN
"no"	CORRECT-PLAN, MODIFY-PLAN
"by the way," "incidentally"	INTRODUCE-PLAN,IDENTIFY-PARAMETER, CORRECT-PLAN
"now," "also"	CONTINUE-PLAN
"anyway"	CONTINUE-PLAN (after popping an interruption)
"ok"	CONTINUE-PLAN, CONTINUE-PLAN (after popping)

This list of clue words would be made available to a parser, which could then note the occurrence of clue words in the input. Thus, if an utterance begins with "by the way," this fact would be available at the start of the plan recognition process. The plan recognition algorithm described in the last chapter can then be explicitly modified as a result. Since the plan recognizer knows that "by the way" is a signal for an interruption, the search will not even attempt to satisfy the first preference heuristic since a signal for satisfaction of the second or third is explicitly present. Without the explicit preface to the utterance, such a focus shift would be unexpected and less coherent, hence the ordering of interruptions as the later preferences. Now we can see that the ordering of the preferences encodes expectations as to the most coherent focus shifts, where shifting of focus has been extended to include interruptions. In particular, the most coherent continuation simply follows the task structure in the top plan, corresponding to preference (1). Following that, a clarification or correction of some completed subplan is expected, corresponding to preference (2). Finally, a topic change (preference (3)) is least expected in the unmarked case.

Consider the following excerpt:

L: The eggplant has been sliced. It's (good) that you advised cutting by judgement instead of absolute directions. We got a monster eggplant that split into ten sections. By the way, the eggplant is turning brown. The traditional method for preventing oxidation is to sprinkle the food with lemon juice. Do you recommend doing so?

M: I'm not sure that it's necessary since we're going to use it soon. If you would like to, you can, but the lemon taste may carry over.

L: I dig. Well skip it then.

M: *By the way*, if you would like some rice, may I recommend you start that soon? (Like now)

L: That's helpful advice, but we don't have any rice.

M: Are you going to use the wok?

Note how "by the way" is used before introducing a new topic of making rice. Without this marker, the plan recognizer would have tried to first interpret making rice as relating to the main dish discussion (which was itself interrupted with "by the way" for the oxidation subdialogue). Thus, if an interruption is explicitly marked the second and third preferences increase priority, so clue words are used to explicitly overrule as well as reinforce default processing strategies. This is in contrast to the last utterance, which is an unmarked topic pop back to the original recipe instructions. Without a discourse marker, the plan recognizer had to first fail in relating using a wok to not cooking rice, before it could recognize the pop. Thus, if unmarked it is a possible interpretation, although hard to find. This is a more flexible model than those of Sidner [90], Grosz and Sidner [40], and Cohen [24], which are unable to handle many unmarked deviations.

4. Incomplete Input

Hendrix [44], Waltz [94], Grosz [37], Carbonell and Hayes [19], and Weischedel and Sondheimer [95] all noted that a large number of elliptical utterances could be understood by modifying a previous utterance based on syntactic and semantic knowledge. For example, Hendrix [44] used a semantic grammar as a basis for finding substitutions. In an input such as

What is the salary of Martin Devin?

Of professors in the Math Department?

"professors in the Math Department" would be parsed as an instance of the category "employee." The previous parse tree would then be used to provide the missing information of this elliptical utterance, by substituting "of professors in the Math Department" for "of Martin Devin," where "Martin Devin" is the old instance of the category "employee."

Unfortunately, none of the above approaches handle cases where the missing portions of an utterance refer to a speaker's non-linguistic, as opposed to discourse, context. This is true when a dialogue begins with an utterance such as a sentence fragment. For example, Allen and Perrault [3] show how a fragment such as "The train to Windsor" could be recognized as a request for the train's time or gate, or as a request for its price, depending on whether the speaker's goal was to board the train or buy a ticket. This need for a plan context also arises when processing certain intersentential fragments. For example, Carberry [17] develops a mechanism using both discourse and task goals to process exchanges such as "I want to register for a course. But I missed pre-registration. The cost?"

The plan recognition theory developed in the last chapter will enable the handling of what will be called *plan ellipsis*, i.e. utterances that contain missing parts that are recoverable from an underlying plan. Such utterances can appear as both sentence fragments and more traditional looking elliptical utterances. For example, imagine an exchange such as

User: Could you possibly retrieve Filename1 and Filename2?

System: Tape numbers?

The system's utterance appears elliptical; however, reformulation of the semantic or case output of the user's utterance will not provide the missing information. One needs knowledge of the underlying plan to do this, e.g. knowledge that one way of retrieving files is by mounting tapes they have been stored on. As will be seen in the examples in the following chapters,

both fragments and elliptical utterances are sufficient input to the plan recognizer. Any missing portions will be recovered as a side-effect of the plan recognition process. This is similar to the treatment of other phenomena governed by immediate focus, i.e. by a local discourse context. That is, ellipsis is filled in without resorting to the plan structure if possible. In such cases, the plan recognizer is given more information with which to begin, and thus has a more constrained search. However, if for some reason a parser is either not augmented with such capabilities, or handling the ellipsis required knowledge of the plan, the plan recognizer will still be able to proceed. The process will be less constrained and more difficult, but possible. Furthermore, once done, the ellipsis will be filled in as a side-effect.

5. Coherence

In order to fully understand a sequence of utterances, one must know how they *cohere*, i.e. one must be able to find implicit relationships between them. For example,

> The text "John took a train from Paris to Istanbul. He likes spinach." is not coherent, even though "he" can refer only to John. At this point the reader may object, "Well, maybe the French spinach crop failed and Turkey is the only country...." But the very fact that one is driven to such explanations indicates that some desire for coherence is operating, which is deeper than the notion of a discourse just being "about" some set of entities. (Hobbs [46])

Current computational models of discourse vary both in the number of relationships allowed, and in the way the relationships can be combined with one another. Nodes in the structures of both Grosz [37] and Cohen [24] are connected by a single type of link (subtask and evidence, respectively). Furthermore, discussion of any given node can only be followed by discussion of nodes produced by certain types of traversals of the structures (unless of course linguistically marked to the contrary). In Grosz [37], discussion of a subtask is generally followed by discussion of the next subtask to be executed. In contrast, McKeown [64], Reichman [76], Hobbs [46], Mann [60], Polanyi and Scha [70], and Woolf and McDonald [99] each have a set of predicates, combinations of which may be captured in a grammar [60, 64, 70, 76, 99]. For example, in the following debate (Reichman [76]), a claim (lines 6-7) is

preceded by an *authority support* (lines 1-4) for the claim.

R: 1. Except however, John and I just saw this two hour TV
2. show

M: 3. Uh hum,

R: 4. where they showed--it was an excellent French TV
5. documentary--and they showed that, in fact, the
6. aggressive nature of the child is not really that
7. much influenced by his environment.

Only Reichman [76], Polanyi and Scha [72] and Grosz and Sidner [40] allow a relationship of interruption, or non-coherency, at any point. Reichman also tries to formalize her conversational moves using a representation quite analogous to that of plan schemas (although her primitives are extremely informal). For example, moves are characterized in terms of their preconditions (discourse context that must be present for appropriate performance), effects on the discourse structure (shifts and status reassignment, expectations) and modes of fulfillment.

The approach taken here lies somewhere in between. Within a single plan, e.g one element of the plan stack, all nodes of the plan's tree structure are connected by subtask links. However, plans are also connected with other plans on the stack, in ways captured by the various meta-plan relationships. If plan execution goes smoothly, subsequent utterances will just *continue* plan execution by shifting to the next subplan. (As mentioned earlier, within each element of the stack of plans, this traversal process can itself be modeled as a stack). However, at any point in the dialogue an utterance can also *interrupt* a plan's execution, for example by *clarifying* which objects are needed for execution or by *debugging* plans that are not executing correctly.

In other words, the implicit relationships allowed in this system are captured by the small set of domain independent meta-plans. They are similar in quantity to the various systems of rhetorical predicates, but differ in quality in that the relationships are all explicitly plan-based. Thus, besides allowing for specification of fairly objective semantics, the same process of plan recognition can be used for both domain and meta-plans, providing a tractable algorithm for

computing the relationships. Furthermore, although the meta-plans are correlates of communicative or rhetorical plans, there is no reason why they could not be used in non-linguistic planning situations as well. Finally, the way the various relationships interact is very flexible (yet simple) and does not require a restrictive grammar. A dialogue is continued (as in Grosz [37] and Cohen [24]) except when interrupted, which can happen at any point. This is similar to Polanyi and Scha [71], where discourse units are created, continued, interrupted, resumed, and completed. With respect to interruptions, we can see how the meta-plans and stack mechanism capture Reichman's manipulation of the context space hierarchies for topic suspension and resumption. If the plan recognized is already on the stack, then the speaker is continuing the current topic, or is resuming a previous (stacked) topic. If the plan is a clarification or correction meta-plan to a stacked plan, then the speaker is commenting on the current topic, or on a previous topic that is implicitly resumed. In other cases, the speaker is introducing a new topic.

6. Interaction of Discourse and Plan Analysis

Since some of the discourse analysis results can be obtained independently of (as opposed to being incorporated into) the plan recognition process, a model of interaction is necessary. Conceptually, the discourse and plan analysis cannot be performed in a fixed sequence. When the task structure is used to guide the discourse structure, which is then used for noun phrase resolution (Grosz [37]), plan recognition (production of the task structure) must be performed first. Similarly, such a priority will be needed in cases where linguistic expectations are violated, such as when questions are ignored. However, suppose the user suddenly changes task plans. Discourse analysis could pick up any clue words signaling this unexpected topic shift, indicating the expectation changes to the plan recognizer. What is important is that either type of analysis should be able to guide the other depending on the utterance, in contrast to any a priori sequential (or even cascaded [11]) ordering. The examples in the following chapters will illustrate the necessity of such a model of interaction.

For the purposes of the current work, a simplified control structure has proved sufficient. Particular portions of the discourse analysis are co-routined with certain portions of the plan recognition process. A subset of the discourse analysis would be performed immediately, in particular clue words are noted and surface phenomena governed by immediate focus are resolved. When the plan recognizer receives these results (along with the other input), it constrains its default processing accordingly. For example, depending on the clue words, the preference ordering of the coherence heuristics might be modified, as when "by the way" overrules the first preference. Or, candidate plans are pruned if the pronoun cannot be resolved as the linguistic analysis suggests. However, without the discourse clues the plan recognizer can still act, although in a much less constrained search space. The plan recognition process will then be used to limit the search space during definite noun phrase resolution, which itself may then provide feedback to the plan recognition algorithm if the resolution fails in the suggested context.

Although such an ordering appears sufficient when discourse markers are used correctly, the ordering may be invalid if the discourse and intentional analyses yield contradictory results. For example, if "by the way" preceded a non-interruption, the clue word analysis would take priority. Thus, an interpretation viewing the utterances as an interruption would be favored over the other possible solutions (and in particular the correct solution) in the search space. An investigation of the misuse of surface linguistic phenomena with respect to discourse function is beyond the scope of this work.

7. Summary

The plan recognition model described in the last chapter was developed to both incorporate and be constrained by a set of results from the area of discourse analysis. In other words, the domain dependent intentional approach was extended to include domain independent knowledge about communication. Such an integration recognizes that a dialogue can be

analyzed along several dimensions. Besides the fact that dialogues have common topics or domains of discourse (here, the tasks being executed), dialogues also cohere due to domain independent strategies for talking about any topic. In this work the former point is reflected in the fact that a chain of meta-plans ultimately referring to a domain plan constitutes a topic. The latter point is reflected in the fact that without linguistic evidence to the contrary, a given plan can only be either continued or interrupted in well-specified ways. Surface linguistic phenomena may add to the coherence of a text by explicitly marking these implicit relationships.

This chapter, in combination with the last, thus presents one way of merging discourse and plan analysis. The simple, more syntactic results of discourse analysis (clue words, immediate focus, ellipsis) are left intact. If such phenomena are present in an utterance, the discourse information they convey is used as input to a plan recognition system, constraining its processing and making it more efficient. Yet, since the plan recognizer is defined independently, when such clues are not present the system can fall back on a purely intentional analysis. This implies, for example, that without such clues, interruptions are hard to process, but not (as others have claimed) impossible. The plan recognizer, in turn, then helps with the resolution of definite noun phrases and plan ellipsis.

In contrast, the more complex semantic results, e.g. rhetorical predicates and the like, have been reformulated in domain independent, meta-planning terms. Their importance in this theory is reflected by the fact that both meta and object plans, e.g goals as well as their relationships with other goals (or at least the fact that there is a relationship, even if no further delineated between a continuation or interruption), must be explicitly recovered when understanding a dialogue. The formulation of such relationships as meta-plans allows more objective semantics as well as use of the tractable plan recognition process. Using this theory, a wide range of subdialogues, including interruptions, can be treated.

Chapter 4

Interrupting Subdialogues

1. Introduction

This chapter and the next will illustrate how the theory developed in the last two chapters is actually applied to discourse phenomena problematic for most systems. In particular, this chapter will show how the system processes dialogues containing interruptions, illustrating its points and generality on two dialogues from different domains and genres.

Each example will show the discourse and plan analysis performed, as well as their interactions. The examples will simulate a trace through a run of the system as it understands the user's utterances in the course of a complete dialogue. The system at present does not concern itself with the planning or generation of natural language responses. The examples will describe what the system should do (using the actual response in the dialogue as a guide), concentrating on how the response effects the representation and analysis of the mutual plan stack.

A computer trace of a slightly simplified version of the "Eight fifty to Montreal" (Litman and Allen [57]) illustrating recognition of a clarification meta-plan and object plan will be presented in Chapter 6. The plan recognition system is implemented on a Vax 750 in Franz Lisp, using the HORNE Reasoning System [6], a lisp-embedded horn clause theorem prover with typing and equality reasoning, as the starting point for the knowledge retrieval mechanism. Aspects of the HORNE typing system were discussed in Section 2.1.1 of Chapter 2 and

need to be elaborated on in order to understand the examples. As we will see, the complex matching supported by unification is very useful when dealing with such frame-like types.

Recall that HORNE types can have a set of distinguished function names called roles, each with an associated type. For example, the type TrainType is a subtype of the type PhysicalObjectType and has three distinguished, type-restricted, roles, as shown:

```
(subtype TrainType PhysicalObjectType
        (gate LocationType)
        (station CityType
        (time TimeType))
```

In other words, role function names are just another type of object to HORNE. The notation ?fn(train1) will be used to represent a role value of train1 (where train1 is an object of type TrainType). Thus, ?fn could be any of the three role names listed above (e.g., gate). If the system later were to match ?fn(train) with a variable of type CityType, ?fn(train1) would be further restricted to the station of train1.

Objects of type *proposition* and *action* are also represented in the type hierarchy and have roles defined for each argument, as is done in semantic network representations. Thus, in the implementation, PARAMETER (term, proposition) is defined to be true only if *term* fills a role of *proposition*.

Finally, the current system assumes that a highly-specialized semantic grammar [12] has parsed the utterances in the train domain. This allows the avoidance of some difficult parsing issues and concentration on the plan recognition model. While such a grammar has in actuality been implemented, it is currently not hooked up to the plan recognition system.

Implementation details concerning issues of knowledge representation will also be glossed over in this chapter. Chapter 6 discusses the requirements a plan recognition system places on a reasoning system and shows how the particular implementation satisfies these requirements.

2. Clarification Subdialogues

This section simulates the system's processing of Dialogue 1, repeated below for convenience.

1) Passenger: The eight-fifty to Montreal?

2) Clerk: Eight-fifty to Montreal. Gate seven.

3) Passenger: Where is it?

4) Clerk: Down this way to the left. Second one on the left.

5) Passenger: OK. Thank you.

The example will show how the system, taking the role of the clerk, understands the passenger's utterances.

The initial state of the system consists of a library of meta-plans and domain plans regarding trains, knowledge of the type hierarchy, and the plan recognition algorithm. The subset of the plan libraries used in this example are repeated below.

HEADER:	GOTO(agent, location, time)
EFFECT:	AT(agent, location, time)
HEADER:	MEET(agent, arriveTrain)
DECOMPOSITION:	GOTO(agent, gate(arriveTrain), time(arriveTrain))
HEADER:	BOARD(agent, departTrain)
DECOMPOSITION:	GOTO(agent, gate(departTrain), time(departTrain)) GETON(agent, departTrain)
HEADER:	TAKE-TRAIN-TRIP(agent, departTrain, destination)
DECOMPOSITION:	SELECT-TRAIN(agent, departTrain, departTrainSet) BUY-TICKET(agent, clerk, ticket) BOARD(agent, departTrain)
CONSTRAINTS:	EQUAL(destination, station(departTrain)) EQUAL(destination, station(departTrainSet)) EQUAL(departTrain, object(ticket))

Figure 4.1: Train Domain Plan Schemas (Repeated from Chapter 2)

The following (simulated) parse of "The eight-fifty to Montreal?" is input to the system:

SURFACE-REQUEST (Person1, Clerk1,
INFORMREF(Clerk1, Person1, ?term, EQUAL(?term, ?fn (dtrain1))))

with constraints:
station (dtrain1) = Montreal
time (dtrain1) = eight-fifty

In other words, Person1 is querying the clerk about some (as yet unspecified) term, ?term, that is the value of some role of dtrain1. Dtrain1 is identified as a train from the input since trains are the only objects in the domain that are described using times and cities. If there were other possibilities, other interpretations would need to be constructed as well. Dtrain1 is restricted to a depart train (i.e. Montreal is restricted to a destination) using the preposition "to." Such an analysis is similar to that produced in Allen and Perrault [3] for interrogative sentence fragments. The next chapter deals with a more general treatment of sentence fragments and will

HEADER:	INTRODUCE-PLAN(speaker, hearer, action, plan)
DECOMPOSITION:	REQUEST(speaker, hearer, action)
EFFECTS:	WANT(hearer, plan) NEXT(action, plan)
CONSTRAINTS:	STEP(action, plan) AGENT(action, hearer)
HEADER:	IDENTIFY-PARAMETER(speaker, hearer, parameter, action, plan)
DECOMPOSITION:	INFORMREF(speaker, hearer, term, proposition)
EFFECTS:	NEXT(action, plan) KNOW-PARAMETER(hearer, parameter, action, plan)
CONSTRAINTS:	PARAMETER(parameter, action) STEP(action, plan) PARAMETER(parameter, proposition) PARAMETER(term, proposition) WANT(hearer, plan)

Figure 4.2: Meta-Plan Schemas (Repeated from Chapter 2)

show how the same results will be achieved using a simpler analysis, one that only knows that the input is a definite noun phrase and a question. Finally, there is no input from the discourse analysis, since the utterance contains no clue words or phenomena governed by immediate focus.

Since the stack is empty, the plan recognizer can only construct an analysis corresponding to coherence preference (3), where an entire plan stack is constructed based on the domain-specific expectations that the speaker will try to take or meet a train. From the SURFACE-REQUEST, via REQUEST, chaining via decompositions produces an instantiation of the INTRODUCE-PLAN meta-plan, as shown in Figure 4.3. The INFORMREF action will be referred to using the name "I1." Although chaining could also proceed from the SURFACE-REQUEST via an INFORM (i.e. the utterance could have been an indirect speech act), an INFORM is a decomposition of IDENTIFY-PARAMETER. Since preference (3) favors plan

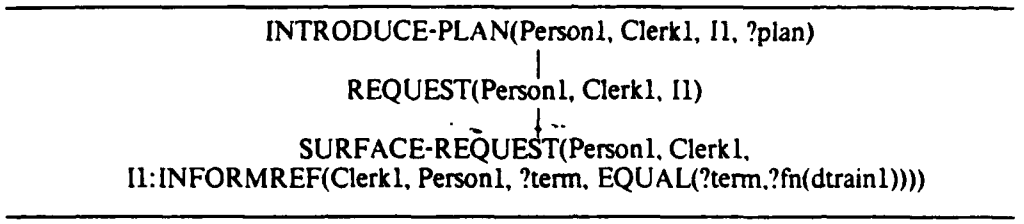


Figure 4.3: Chaining Produces an Intermediate Plan Recognition Structure

introductions, the clarification interpretation is thus eliminated. Similarly, chaining from an alternative SURFACE-REQUEST to INFORMIF parse would also be eliminated since a yes/no question interpretation is inappropriate.

Before pursuing the candidate plan any further, the plan recognizer checks on the plan's reasonableness using the plan-based heuristics. From constraint satisfaction it knows that the INFORMREF must be a step in an object plan. To satisfy this constraint, i.e. STEP(I1,?plan), an object plan will be created (and arbitrarily called PLAN2). This new state of affairs is shown in Figure 4.4, where the name of a plan structure appears at the top left-hand corner. In accordance with the constraint, the INFORMREF I1 is part of PLAN2.

PLAN1

INTRODUCE-PLAN(Person1, Clerk1, I1, PLAN2)

REQUEST(Person1, Clerk1, I1)

SURFACE-REQUEST(Person1, Clerk1,I1)

PLAN2

I1: INFORMREF(Clerk1, Person1, ?term, EQUAL(?term, ?fn(dtrain1)))

Figure 4.4: INTRODUCE-PLAN and its Object Plan

The second constraint, e.g AGENT(I1, Clerk1) is already satisfied. Finally, the recognizer verifies that the effects of the meta-plan are not already true, i.e. that the clerk does not already have PLAN2 as a goal. Since the heuristics support PLAN1, the recognizer can now resume chaining, but since INTRODUCE-PLAN is not a step in any plan, chaining stops.

However, recall that INTRODUCE-PLAN is a meta-plan and via constraint propagation an associated object plan (PLAN2) has been introduced. The recognizer recursively expands this plan and recognizes that it is part of an IDENTIFY-PARAMETER plan, using decomposition chaining. In satisfying the constraints on IDENTIFY-PARAMETER (Clerk1, Person1, ?parameter, ?action, ?plan), i.e.

1. PARAMETER(?parameter,?action)
2. STEP(?action,?plan)
3. PARAMETER(?parameter,EQUAL(?term,?fn(dtrain1)))
4. PARAMETER(?term,EQUAL(?term,?fn(dtrain1)))
5. WANT(Person1,?plan)

a third plan is introduced (constraint (5)) that must have a step (2) that contains a property of a train (1.3), described via the equality of the INFORMREF (4). An eligible plan is GOTO, where ?fn(dtrain1) is restricted to be the time or location of the GOTO. As a result of this, ?fn is restricted to be a time or gate role of dtrain1. Another eligible domain plan is TAKE-

TRAIN-TRIP, where destination is the parameter being identified. These are the only two of the domain and meta-plan schemas satisfying all the constraints, in particular the constraint that the plan schema contains a location, city or time (i.e. a role value of dtrain1) as a parameter.[1] Both the identification of the destination in TAKE-TRAIN-TRIP and the time in GOTO are eliminated by the plan heuristic that one does not execute plans when its effects are already true. In this case, the destination and time of dtrain1 were specified in the parse and are thus already known. In other words, since the effect of the IDENTIFY-PARAMETER plan needs to be achieved, the speaker does not know enough about some property of the train to execute, for example, GO1. Since the time was known from the utterance, the role name time can be eliminated as a possibility for the value of ?fn due to the heuristic that one does not need to execute a plan if the effect is already true. The plan recognition heuristics discussed above thus eliminate competing interpretations and constrain the object plan to GOTO and ?fn to be the gate. Figure 4.5 show the three plan structures created so far.

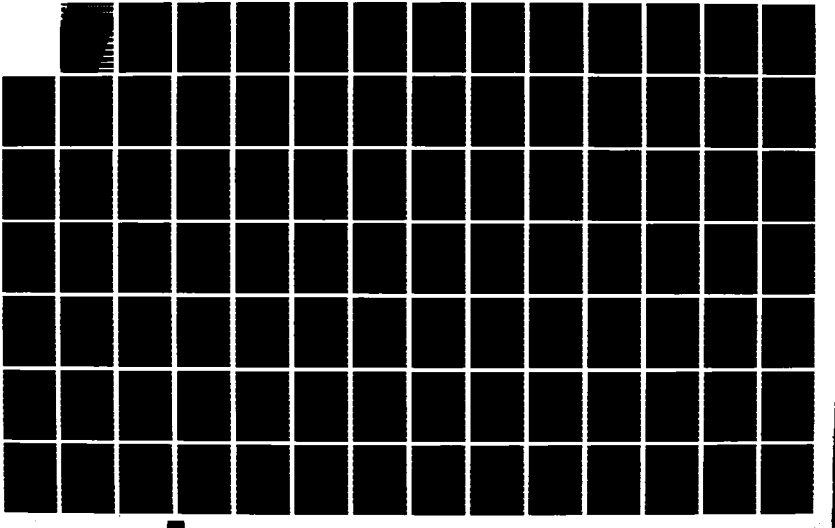
Chaining from PLAN2 produces no higher level goals, so the plan recognition algorithm is recursively called on that plan's object plan, here PLAN3. Decomposition chaining from GO1 yields both the BOARD and MEET plans. The MEET plan is eliminated due to typing constraint violation; dtrain1 is already known to be a departing train from the parse. Also, since the expected agent of the BOARD plan is the speaker, ?agent is set equal to Person1. Finally, chaining proceeds from BOARD to TAKE-TRAIN-TRIP.

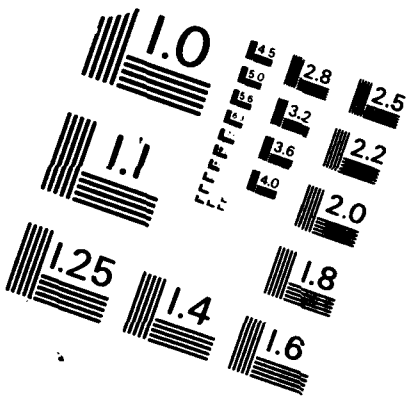
Once the recursive call is completed, plan recognition ends, all postulated plans are expanded top down to include the rest of their steps, and the stack is constructed. Note that all three plans are recognized before any is placed on the stack. Furthermore, although GO1O for example, is recognized after IDENTIFY-PARAMETER, it is put on the stack first since it

[1]The reader may be wondering why buying a ticket was not recognized as a possible object plan. This was due to the simplified treatment of sentence fragments for the purposes of this chapter. For example, noun phrases can indicate not only questions about the noun phrase's role (as above), but also questions about roles of objects for which the noun phrase itself is a role. Thus, "The 8:50 to Montreal" could also be used to refer to another role of its associated ticket, for example the cost. Since BUY-TICKET would eventually be eliminated due to constraint violation (i.e.

AD-A170 871 PLAN RECOGNITION AND DISCOURSE ANALYSIS: AN INTEGRATED APPROACH FOR UNDERSTANDING DIALOGUES(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE D J LITMAN 1985 TR-170 2/3

UNCLASSIFIED N00014-82-K-0193 F/G 5/7 NL





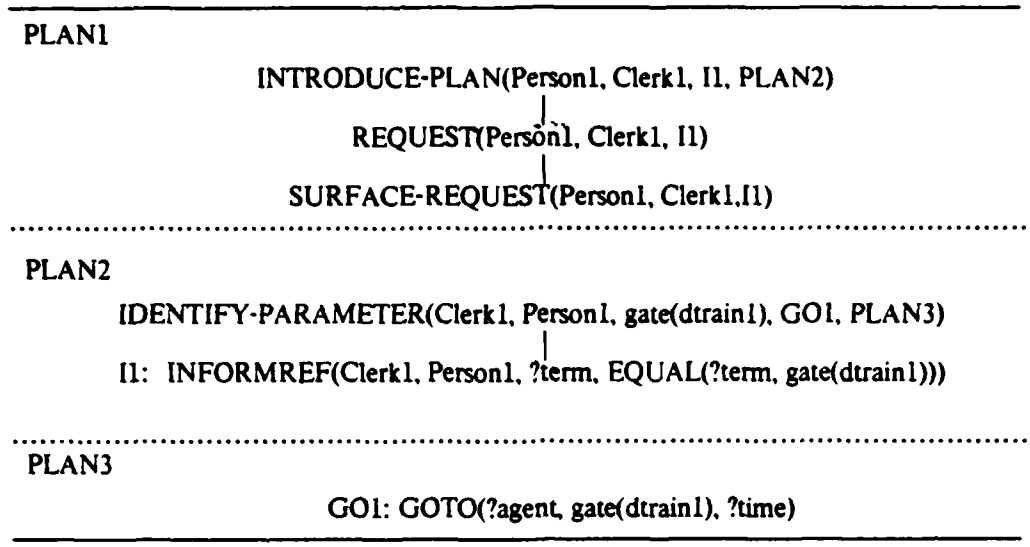


Figure 4.5: Constraint Satisfaction Creates PLAN2 and PLAN3

is suspended for the IDENTIFY-PARAMETER clarification. The state of the stack after the plan recognition process just discussed is shown in Figure 4.6, which should be viewed as one stack with three elements: PLAN1 is at the top of the stack, PLAN2 in the middle, and PLAN3 at the bottom. The changes from the previous figure are italicized; the dotted lines indicate the information known from the top down expansions. As desired, we have constructed an entire plan stack based on the original domain-specific expectations that the speaker (although more generally it could be someone besides the speaker) will take or meet a train.

Once the task structure is recognized the focus (the executing step) in each plan structure is noted, as shown in square brackets in the figures. Thus the SURFACE-REQUEST in PLAN1 is marked as the LAST (the executed) step of PLAN1, which is now completed. From the effects of INTRODUCE-PLAN and IDENTIFY-PARAMETER we mark I1 and GO1, respectively, as in focus whenever execution returns to those stacked plans.

the clerk can only give information), the omission is insignificant with respect to the points being illustrated.

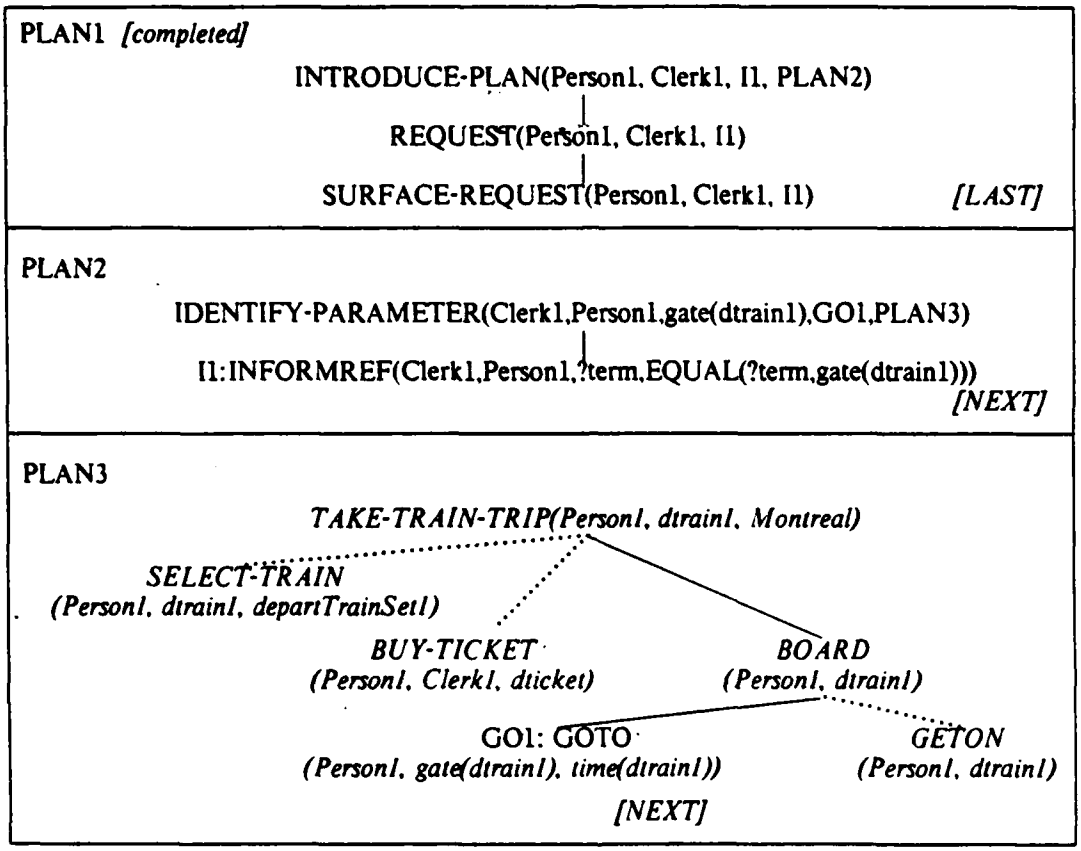


Figure 4.6: The Plan Stack after the First Utterance

The clerk's planning of the response is not specified in this theory. Here what the system should do will be described. The system responds with the expected continuation (and implicit resumption), namely the INFORMREF in PLAN2, popping the completed PLAN1 off the stack. To execute the INFORMREF, the system would plan a SURFACE-INFORM, which might eventually be realized as the utterance "Eight-fifty to Montreal. Gate seven." Using the assumption that this utterance is correctly recognized by the passenger, the system updates the focus in PLAN2, marking the new SURFACE-INFORM step as [LAST] and the plan as [completed]. This anticipates the passenger's correct interpretation of the utterance, since the stack is assumed to be shared between the speakers. Although all the steps of PLAN2 are completed,

its success cannot be confirmed until the next utterance. Thus it is not popped off the stack. This allows the possibility of a clarification plan being introduced concerning PLAN2. Thus, at the stage just before Person1 speaks again, the stack would contain PLAN3 and PLAN2, as shown in Figure 4.7.

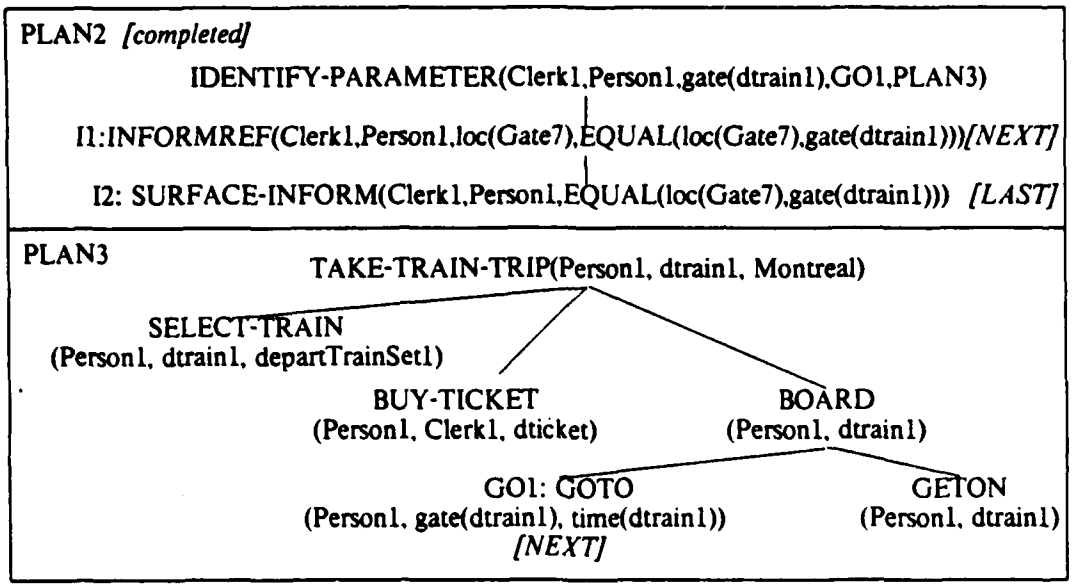


Figure 4.7: The Plan Stack after the Clerk's Response

The system is now in a state to recognize a continuation of PLAN3 (preference (1)) or a meta-discussion of PLAN2 or PLAN3 (preference (2)). Preference (3) would involve abandoning the current stack, so is very unlikely. Meta-discussion of PLAN1 is also very unlikely since it has been popped from the stack, and in fact would require explicit markers for its recognition (for example, "Oops, I meant to say...").

The passenger then asks "Where is it?", which would be parsed into the action:

SURFACE-REQUEST (Person1, Clerk1
INFORMREF (Clerk1,Person1,?term1, EQUAL (?term1,loc(Gate7))))

This analysis assumes the appropriate resolution of "it" to Gate7 by an immediate focus

mechanism such as Sidner's [88]. This makes the example simpler. In this case the plan recognition would work even if the "it" were not resolved and left as an unknown constant, as discussed below.

Although the clerk thought the SURFACE-INFORM of PLAN2 achieved the desired passenger KNOWREF, in actuality it did not provide a description enabling the passenger to execute the GOTO of PLAN3. Instead we have another request for clarification. The plan recognizer attempts to incorporate this utterance based on the coherence heuristics. The first preference fails since the SURFACE-REQUEST does not match (directly or by chaining) any of the steps in PLAN3, which should have been resumed if the previous clarification was understood. This preference, then, involving popping the completed PLAN2, is not possible because the utterance cannot be seen as any step of the TAKE-TRAIN-TRIP plan. The second preference succeeds, and the utterance is recognized as part of an introduction of a new IDENTIFY-PARAMETER referring to the old one. This process is basically analogous to the process discussed in detail above, except that the plan to which the IDENTIFY-PARAMETER refers is found in the stack rather than constructed. The final results of the analysis are pushed onto the stack after popping to the appropriate object plan (which is here already on top), as shown in Figure 4.8. Again, the fact that an interruption occurred, as well as how the interruption is related to the interrupted topic, is eventually recognized without the use of any explicit discourse markers.

As mentioned above, if in the input "it" was originally unresolved (i.e. Gate7 in I3 is replaced with ?pronoun), "it" will be correctly resolved later as a side-effect of the plan recognition. In particular, the appropriate binding will be made during the constraint propagation process connecting PLAN5 to PLAN2.

Using the actual clerk's reply, "Down this way to the left -- second one on the left" as a guide, the clerk's response is simulated as a pop of the completed PLAN4, followed by execu-

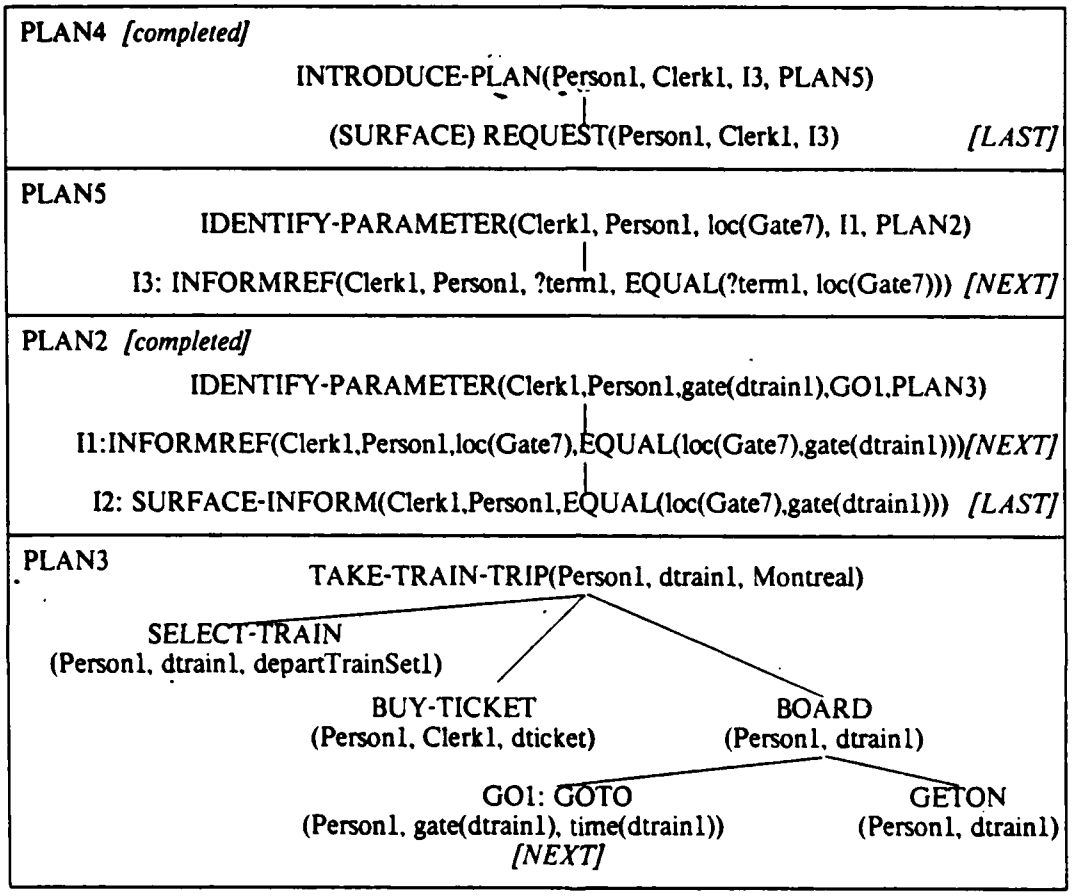


Figure 4.8: The Plan Stack after the Passenger's Second Utterance

tion of the INFORMREF I3 and thus completion of PLAN5. The system is now ready to recognize the passenger's next plan, likely a pop of all completed plans, leading back to TAKE-TRAIN-TRIP (preference 1). Or, less likely, there are several options corresponding to the second preference class. For example, the passenger could execute a meta-plan to the top IDENTIFY-PARAMETER (e.g., "Second what?") or a pop. The pop allows a meta-plan to the stacked IDENTIFY-PARAMETER of PLAN2 ("What's a gate?") or a pop, which allows a meta-plan to the original domain plan ("It's from Toronto?") or a pop. Each of these possible stacks is shown in Figure 4.9. Finally, a totally new topic could be pushed onto the stack, in

P: The eight-fifty to Montreal?
C: Eight-fifty to Montreal. Gate seven.
P: Where is it?
C: Down this way to the left. Second one on the left.

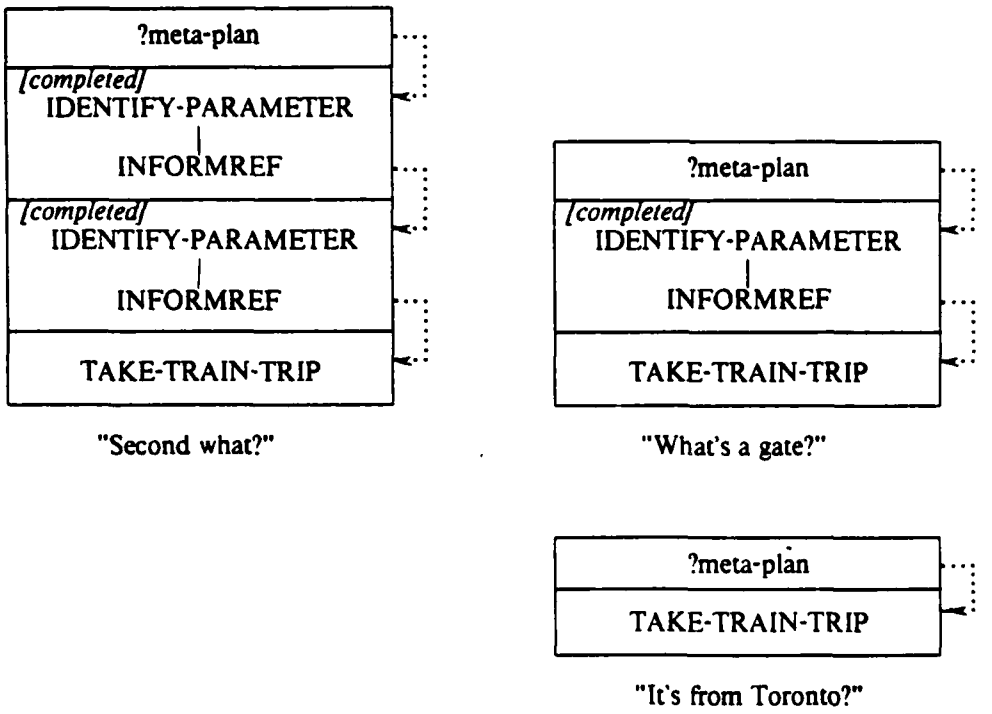


Figure 4.9: Coherent Dialogue Continuations of Preference Two

accordance with preference (3).

The dialogue actually concludes with the first preference, resumption and continuation of TAKE-TRAIN-TRIP. This can be immediately recognized due to the passenger's use of the two discourse clues in "OK. Thank you." Recall that "OK" is an example of a clue word marking CONTINUE-PLAN, possibly after a pop off the stack. This analysis is further reinforced by "Thank you," a discourse convention also indicating some level of plan or subplan completion, as well as termination of the dialogue if not followed by further utterances. Note

that the original domain plan involves no communication, i.e. there are no further utterances that can execute the domain plan. Also note that unlike the previous utterance, what is going on with respect to the plan recognizer is determined solely by the discourse analysis (i.e., intentionally, the utterance is vacuous).

3. Correction Subdialogues

This section presents a hand trace of the processing of Dialogue 2 (repeated below), illustrating not only an interrupting clarification, but also an interruption corresponding to a debugging as well as the resumption and continuation of the original plan.

User: Show me the generic concept called "employee."

System: OK. <system displays network>

User: I can't fit a new ic below it. Can you move it up?

System: Yes. <system displays network>

User: OK, now make an individual employee concept whose first name is "Sam" and whose last name is "Jones." The Social Security number is 234-56-7899.

System: OK.

The section also illustrates the treatment of indirect speech acts, and shows how multiple utterances within a single user turn are understood. Finally, the example shows the domain independence of the approach; although new domain plans must be introduced, the recognition procedure and the meta-plans remain unchanged. Furthermore, since the model applies equally well to task-oriented and information-seeking domains, a bit of genre independence can also be claimed.

Recall that in the above dialogue the user interacts with a KL-ONE database system that is capable of graphically displaying KL-ONE concepts (KL-ONE [11] is a knowledge representation language). Figure 4.10 presents the relevant domain plan schemas for this example. They are taken from Sidner and Israel [86], with minor modifications and consist of plans to add new data into the network and to examine parts of the network. Both of these have a subplan

involving the plan CONSIDER-ASPECT, in which the user considers some aspect of a network, for example by looking at it (the decomposition shown), listening to a description, or thinking about it. Again, the representation of action and time is greatly simplified. The subset of the meta-plan library matched in this example is repeated in Figure 4.11. (As will be seen by referring to Figure 2.5 during the matching attempts, MODIFY-PLAN will not be a viable candidate plan for any utterance).

HEADER:	ADD-DATA(user, netpiece, data, screenLocation)
DECOMPOSITION:	CONSIDER-ASPECT(user, netpiece) PUT(system, data, screenLocation)
HEADER:	EXAMINE(user, netpiece)
DECOMPOSITION:	CONSIDER-ASPECT(user, netpiece)
HEADER:	CONSIDER-ASPECT(user, netpiece)
DECOMPOSITION:	DISPLAY(system, user, netpiece)

Figure 4.10: Graphic Editor Domain Plans

The processing begins with the analysis of "Show me the generic concept called 'employee'."

SURFACE-REQUEST (user, system, DISPLAY (system, user, E1))

where E1 stands for "the generic concept called 'employee.'" The discourse analysis provides no input. Since there is no stack, the plan recognizer tries to introduce a plan (or plans) according to preference (3). Through forward chaining, the system finds that the utterance is a REQUEST that introduces some plan. From constraint satisfaction we know that the plan introduced must contain the display action. Since INTRODUCE-PLAN is not a step in any other plan chaining stops; since it is a meta-plan, recognition from the DISPLAY now occurs. Since the display action could be a step of the CONSIDER-ASPECT plan, which itself could

HEADER:	INTRODUCE-PLAN(speaker, hearer, action, plan)	
DECOMPOSITION:	REQUEST(speaker, hearer, action)	
EFFECTS:	WANT(hearer, plan)	NEXT(action, plan)
CONSTRAINTS:	STEP(action, plan)	AGENT(action, hearer)
HEADER:	CONTINUE-PLAN(speaker, hearer, step, nextstep, plan)	
PREREQUISITES:	LAST(step, plan) WANT(hearer, plan)	
DECOMPOSITION:	REQUEST(speaker, hearer, nextstep)	
EFFECT:	NEXT(nextstep, plan)	
CONSTRAINTS:	STEP(step, plan) STEP(nextstep, plan) AFTER(step, nextstep)	AGENT(nextstep, hearer) CANDO(hearer, nextstep, plan)
HEADER:	IDENTIFY-PARAMETER(speaker, hearer, parameter, action, plan)	
DECOMPOSITION:	INFORMREF(speaker, hearer, term, proposition)	
EFFECTS:	NEXT(action, plan) KNOW-PARAMETER(hearer, parameter, action,plan)	
CONSTRAINTS:	PARAMETER(parameter, action) STEP(action, plan) PARAMETER(parameter, proposition) PARAMETER(term, proposition) WANT(hearer, plan)	
HEADER:	CORRECT-PLAN(speaker, hearer, laststep, newstep, nextstep, plan)	
PREREQUISITES:	WANT(hearer, plan) LAST(laststep, plan)	
DECOMPOSITION-1: DECOMPOSITION-2:	achieve WANT(hearer, newstep) achieve WANT(hearer, nextstep)	
EFFECTS:	STEP(newstep, plan) AFTER(laststep, newstep) AFTER(newstep, nextstep) NEXT(newstep, plan)	
CONSTRAINTS:	STEP(laststep, plan) STEP(nextstep, plan) AFTER(laststep, nextstep) AGENT(newstep, hearer) CANDO(speaker, nextstep, plan) MODIFIES(newstep, laststep) ENABLES(newstep, nextstep)	

Figure 4.11: Meta-plan Schemas used for Dialogue 2 (Repeated from Chapter 2)

be a step of either the ADD-DATA or EXAMINE plans, the domain plan remains ambiguous. Note that heuristics can not eliminate either possibility, since at the beginning of the dialogue any domain plan is a reasonable expectation. Chaining halts at this branch point and all plans are expanded (again shown by dotted lines).

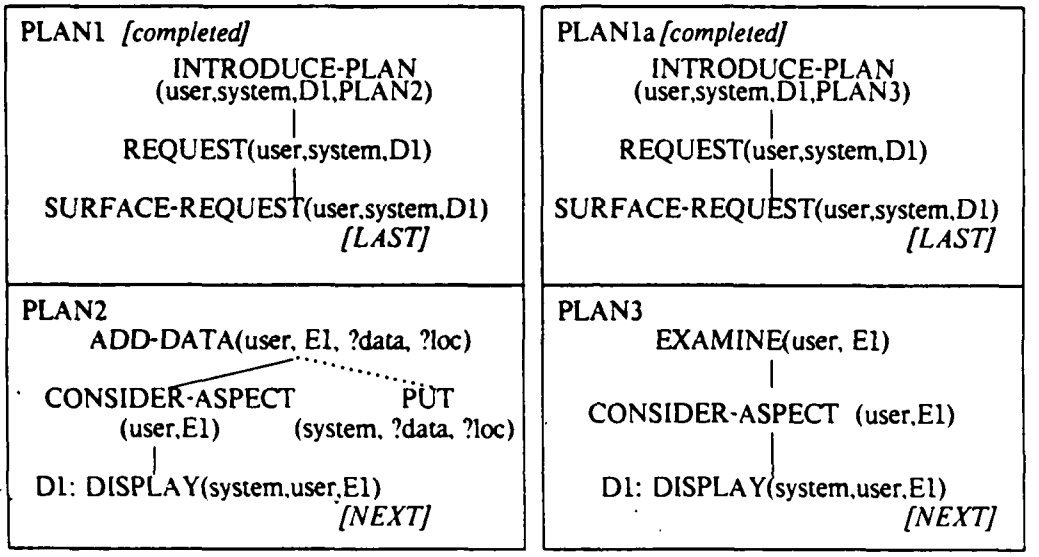


Figure 4.12: The Two Plan Stacks after the First Utterance

When the executed plan structures are recognized, their effects can also be asserted. In particular, the focus can be noted. Since the discourse structure follows the task structure the executed SURFACE-REQUEST is marked as focused and the introduction meta-plans as completed. Since among the effects of any meta-plan is an updating of the focus in the plan referred to, the DISPLAYs of plans 2 and 3 are marked as focused. Finally, two stacks are constructed as shown in Figure 4.12. As desired, we have constructed a plan stack for each interpretation based on domain-specific expectations.

The system, assumed cooperative and a planner itself, examines the postulated plans and decides what to do (again, the planning the system performs is actually simulated based on the dialogue). Since the REQUEST (and thus plan introduction) is completed, in each stack the

stack can be popped and the execution of DISPLAY performed as a coherent next move. The system chooses to perform the display even though the reason for this action is still ambiguous. Deciding exactly what to do in such cases is an interesting planning issue. For example, in this case we will see that although the REQUEST seemed fairly explicit, the system's lack of higher-level knowledge led to a non-optimal response. The system also chooses to generate "OK" to explicitly mark the step's completion.

The user's response, "I can't fit a new IC below it," is input as SURFACE-INFORM (user, system, ~CANDO (user, FIT (user, ?ic[2] , belowE1))), where FIT is an instance of PUT (the terms will be used interchangeably). In other words, the fact that the system decided to perform the DISPLAY without knowing whether a PUT would follow (and if so what would be PUT and where) has now caused problems in the user's original plan, viz., the location of the node for the generic concept did not leave enough room to fit a new IC node below it. The resolution of "it" to E1 is done by the immediate focus and incorporated into the parse as shown.

The utterance is recognized by the plan recognizer as either an indirect REQUEST for the system to perform the FIT or a literal INFORM, and since neither is yet connected to any domain or meta-plan, we pursue both alternatives. (Recall that the incremental plan recognition process holds for plan branch points, but not for speech act recognition.) For each stack, recognition of a continuation of its plan (preference 1) fails. In the first stack, CONTINUE's CANDO(system, FIT(system, ?ic, belowE1, PLAN2) constraint fails. For the stack containing PLAN3, there are not enough steps in PLAN3 to satisfy the conditions of its continuation.

---

[2]This parse assumes that the use of "a new IC" is *attributive* (Donnellan [28]), i.e. the user is stating that any new ic will not fit below E1. This is in contrast to a *referential* use, where the user has a particular ic in mind. While these terms were developed for definite descriptions, similar ideas are useful for indefinite descriptions as well. For example, in "Could you mount a magtape for me? It's tape 1" the speaker obviously has a particular magtape in mind when "a magtape" is uttered. In the next chapters we will see how such indefinite expressions are represented as skolem functions rather than variables. For the current purposes the distinction is irrelevant.

Chaining from the indirect REQUEST(user, system, F1: FIT(system, ?ic, belowE1)) (recall Figure 2.8) we may also postulate that either of the two possible plans is being corrected, i.e., a meta-plan to one of the stacked plans is introduced (preference (2)). Since the REQUEST matches both decompositions, there are two possibilities for CORRECT-PLAN:

CORRECT-PLAN(user,system,?laststep,F1,?nextstep,?plan)

CORRECT-PLAN(user,system,?laststep,?newstep,F1,?plan)

where the parameters of each will be bound, or at least further restricted, as a result of constraint and precondition satisfaction from application of the heuristics. For example, recall that candidate plans are only reasonable if their preconditions were true, i.e. (in both stacks and corrections) WANT(system, ?plan) and LAST(?laststep, ?plan). Assuming the plan was executed in the context of PLAN2 or PLAN3 after the DISPLAY was performed, ?plan could only have been bound to PLAN2 or PLAN3, and ?laststep bound to D1. Then, satisfaction of the constraints will eliminate the PLAN3 binding, since the constraints indicate at least two steps must be in the plan, while PLAN3 contains no sequence of steps (just a single step described at different levels of abstraction). Finally, with respect to the effects not being already true for each hypothesis, the first effect yields STEP(F1,PLAN2) and STEP(?newstep, PLAN2), respectively. Since the first is already true, with respect to PLAN2 the first CORRECT-PLAN is eliminated. Thus only the second correction on the first stack shown in Figure 4.12 remains plausible, and in fact, using PLAN2 and the latter correction the constraints can be satisfied. In particular, the bindings so far yield:

(1) STEP(D1, PLAN2)
(2) STEP(F1, PLAN2)
(3) AFTER(D1, F1)
(4) AGENT(?newstep, system)
(5) ~CANDO(user, F1, PLAN2)
(6) MODIFIES(?newstep, D1)
(7) ENABLES(?newstep, F1)

Constraint (1) is already valid, and (2), (3), and (5) made valid, assuming the matching of PUT and FIT1 (which was implied in the above discussion). Finally, assuming ?newstep can still not

be uniquely determined, it is now further constrained to satisfy (4), (6), and (7).

The effects of this new plan are asserted (?newstep is inserted into PLAN2 and marked as in focus), and CORRECT-PLAN is pushed on to the stack on top of its object plan, as shown in Figure 4.13. We have thus recognized not only that an interruption has occurred, but also how the interruption is related to the interrupted topic. Finally, note that since PLAN4 contains an unbound parameter it is not yet marked as completed.

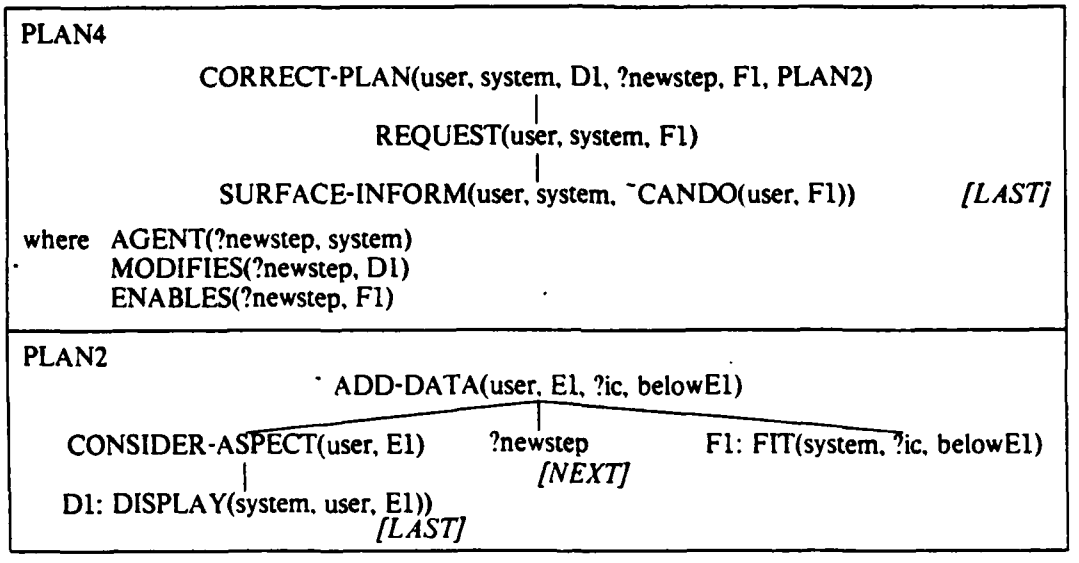


Figure 4.13: The Plan Stack after the User's Second Utterance

Any other preference (2) possibilities (a REQUEST or INFORM leading to a modification or a clarification) fail due to constraint violation. Finally, a REQUEST for introduction of a new plan is discarded since it does not tie in with any of the expectations (i.e. a preference (2) choice is preferred over a preference (3) choice).

The analysis of the user's continuation, "Can you move it up?" is

SURFACE-REQUEST (user, system, INFORMIF (system, user
(CANDO (system, M1: MOVE (system, E1, up))))),

(with E1 again from the immediate focus), recognized as either a literal or indirect REQUEST.

The first preference does not apply; PLAN4 has no steps to CONTINUE, but because it is not complete PLAN2 shouldn't be resumed and continued. Within preference (2), from the indirect REQUEST we can chain via its effect KNOW(system, WANT(user, M1)) to INFORMREF(user, system, M1, WANT(user, M1)). From the INFORMREF we can chain to IDENTIFY-PARAMETER, which can be related to PLAN4 via satisfaction of the constraints of IDENTIFY-PARAMETER as well as the listed constraints for PLAN4 in Figure 4.13. (The other possible INFORMREF, INFORMREF(user, system, user, WANT(user, M1)) and its IDENTIFY-PARAMETER are eliminated since the effect is already true, i.e. the system already knows the person with the goals is the user.) Note that if E1 in M1 was not known (i.e. the pronoun "it" in "Can you move it up" was not resolved to E1), the resolution could now be made purely intentionally via satisfaction of PLAN4's three constraints on MOVE (although since the constraints are now less specified, proving their satisfaction is harder than in the earlier case). The effects of the recognized plan are asserted, in particular M1 is bound to ?newstep in both PLAN4 and PLAN2, and the plan is then pushed onto the stack, as shown in Figure 4.14. PLAN4 can now be considered completed since it is fully instantiated by the effects of PLAN5, and its steps all performed. (Note that if the system had been able to infer ?newstep before this utterance, PLAN5 would have seemed somewhat redundant (although in actuality it would involve making distinctions between belief and knowledge) and thus would have been harder to recognize.)

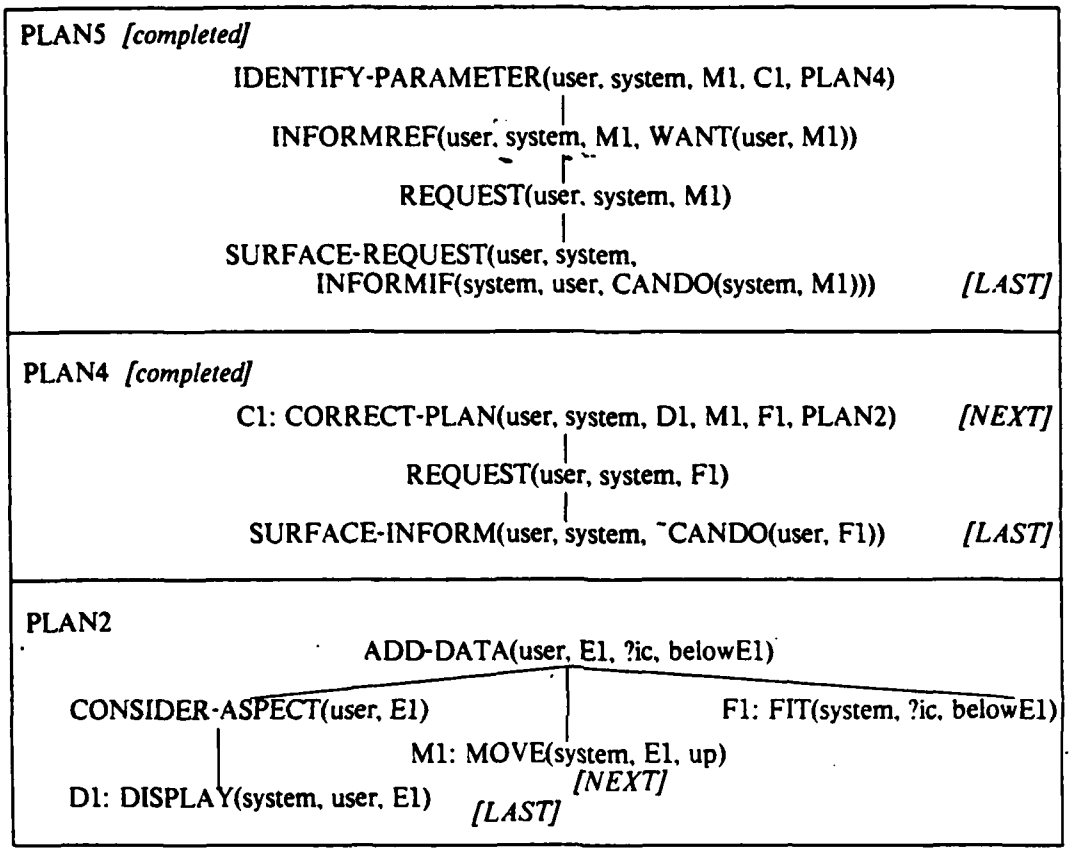


Figure 4.14: The Plan Stack after the User's Elaboration

The other IDENTIFY-PARAMETER act, obtained by chaining from the literal REQUEST, cannot tie into PLAN4 so is eliminated. Since CORRECT-PLAN specifically had a parameter to be identified, any other possible preference (2) meta-plans to PLAN4 would only be considered if IDENTIFY-PARAMETER couldn't be recognized.

This latter point illustrates some perspectives suggested by this theory on recognition of *narrative text*, sequences of utterances by a single speaker. For example, in cases where a plan is incompletely recognized although all of its steps have been executed, there is a strong expectation for a completion of the details by the same speaker. Furthermore, such "elaborations" (to use a term commonly seen as a coherence relation) are seen as yet another topic

suspension, analogously to the cases described above. While perhaps unintuitive (i.e. a plan is interrupted for, rather than continued by, an elaboration), it enables the recognizer to deal with narrative using only already existing structures and mechanism.

Back to the stack of Figure 4.14, the system is simulated as follows. Noting that the user's IDENTIFY-PARAMETER and CORRECT-PLAN are both complete, it pops the stack and resumes PLAN2 with the new step M1 inserted by the meta-plans. Once M1 is done, the focus is updated and the system is now ready to recognize the continuation of the plan with F1.

Before the parse of the user's next utterance ("OK, now make ...") is even processed, the system's discourse mechanism would pick up the two initial clue words that explicitly mark this continuation with the next step, thus reinforcing the coherence preference (1). The rest of the utterance is then recognized as a continuation meta-plan (analogously to the above detailed explanation, where M1 and PLAN2 are bound due to precondition satisfaction, and MAKE1 chained through FIT due to constraint satisfaction (recall Figure 2.9)), as shown in Figure 4.15. At this stage, it would then be appropriate for the system to pop the completed CONTINUE plan and resume execution of PLAN2 by performing MAKE1.

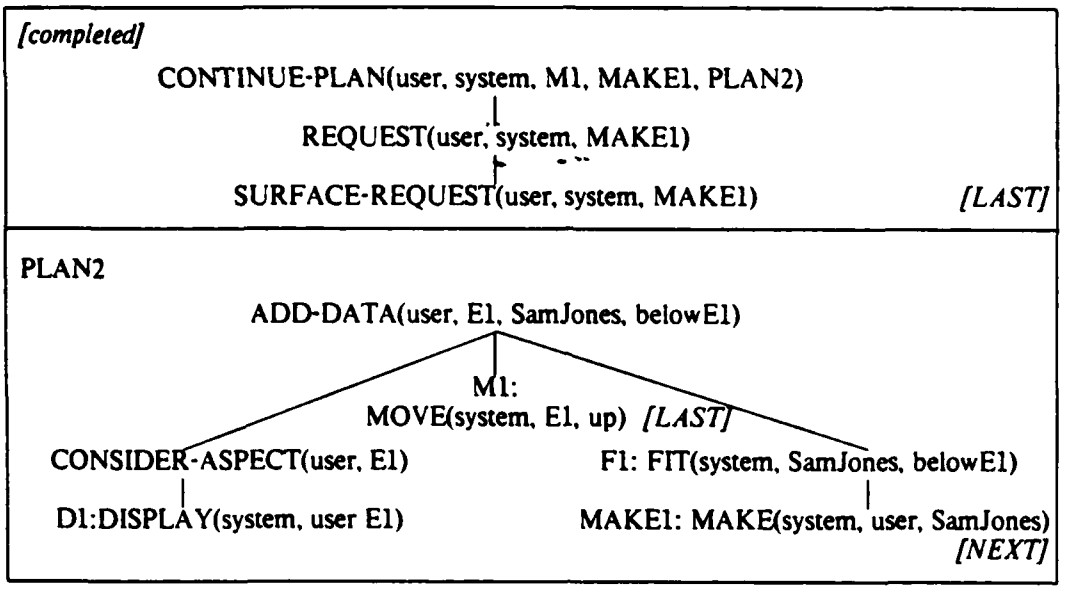


Figure 4.15: Continuation of the Original Domain Plan

Note how the framework would still prove useful if the user's "Can you move it up?" was preceded by the system's "What do you want me to do?". In other words, assuming the user did not provide the expected elaboration the system could plan to ask for it. The analysis is shown in Figure 4.16. Comparing this with the situation in Figure 4.14, note that after introduction by PLAN6 the analysis will be the same as for the non-prompted case.

It is also interesting to note what kind of analysis could result if the user's second and third utterances were reversed. Assume the user says "Can you move it up?" in the plan recognizer's context of Figure 4.12 (although after the display has been executed). As before, the utterance can not be recognized as a continuation, expected since the system believes it acted correctly. CORRECT-PLAN (user, system, D1, M1, FIT, PLAN2) is recognized and this plan is put above its object plan, PLAN2, on the stack (as shown by looking at only the bottom two elements of Figure 4.17, with the FIT not yet fully instantiated). Notice that the system knows both what to do and why, since from the constraints the system can infer both that

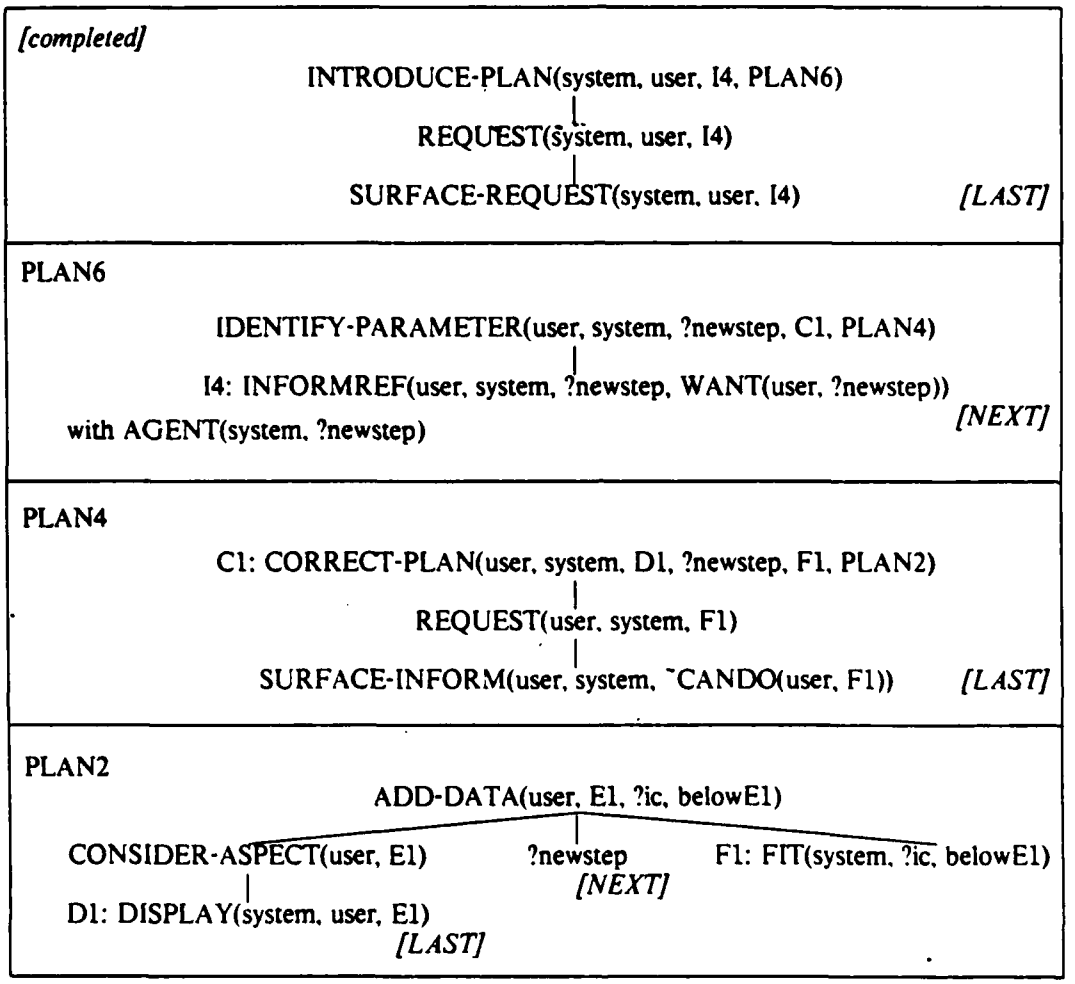


Figure 4.16: If the System had Prompted

?newstep is bound to M1 and that the user can't do the FIT, i.e. that the interaction between DISPLAY (?laststep) and FIT (?nextstep) is the problem. This was not true when CORRECT-PLAN was first recognized in the original dialogue (Figure 4.13), where the problem (F1) was explicitly given, but the solution (?newstep) could only be partially restricted (as opposed to uniquely identified). The user explicitly confirms this with the next utterance, "I can't fit a new ic below it," recognized as an IDENTIFICATION of the FIT parameter in CORRECT-PLAN, which fills in the previously unknown parameters of the FIT. As above, since there were

unknown parameters such an elaboration would be expected (and note that the dialogue sounds somewhat incomplete without it, although as has been pointed out to me one would not expect to have to explain motivations in a dialogue with a real machine). The clarification is then pushed on the stack; the state of the stack after the two utterances would be as shown in Figure 4.17. Comparison of Figures 4.14 and 4.17 illustrate how the same utterances are analyzed differently depending on the discourse context. In particular, although in each case the system recognizes an interruption of the CORRECT-PLAN, the particular relationship between the interruption and the CORRECT-PLAN is different.

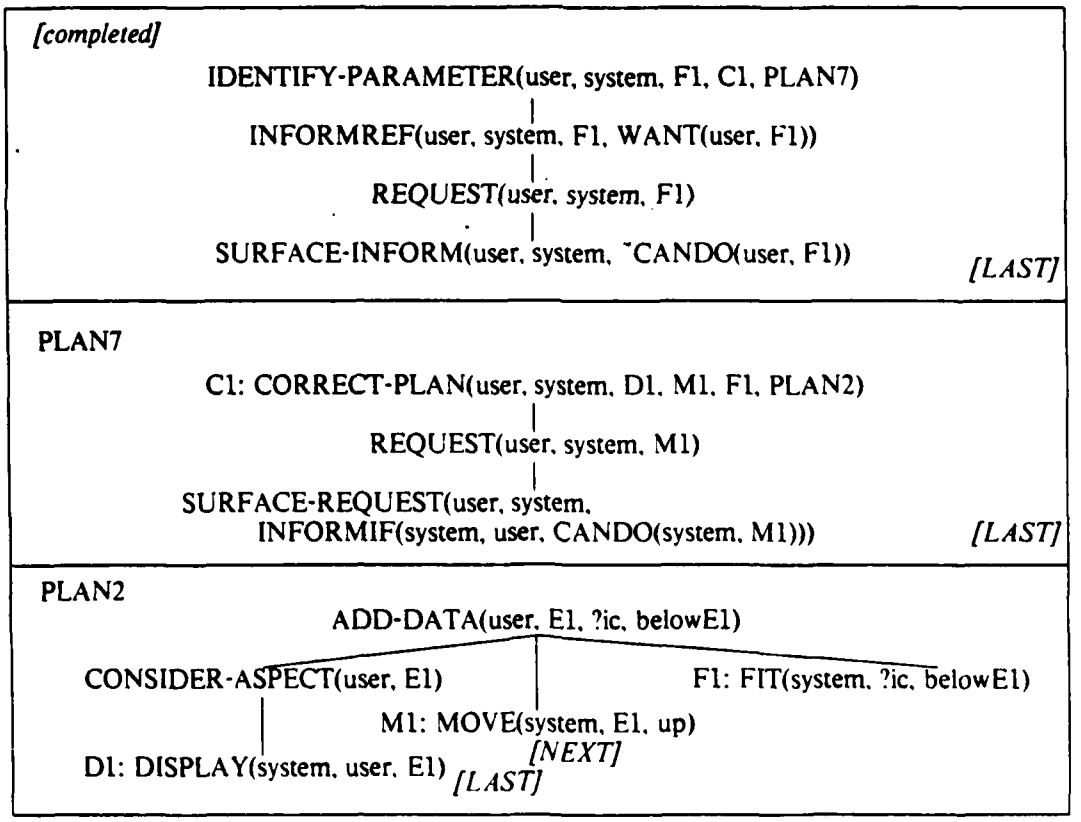


Figure 4.17: If the User's Utterances were Reversed

4. Summary

This chapter has illustrated how the theory developed in the last two chapters is actually used to process specific dialogues, using dialogues from different domains and genres to support the generality of the theory. By tracing through the plan recognition algorithm in detail, we can see exactly how and when the results of the discourse analysis as well as the structures in the plan libraries are used to create and update a stack (or stacks) of postulated plan structures. We can also see exactly how the various heuristics can be used to effectively prune many of these possibilities, and how meta-plans are connected to object plans (as well as how the appropriate object plan is found). At the beginning of a dialogue the process of constraint satisfaction will cause the actual construction of an object plan satisfying the meta plan's requirements. During a dialogue, the constraint satisfaction process will be used to connect new meta-plans to existing plan structures on the stack.

This chapter has thus shown how the theory handles a range of discourse phenomena. The interpretation of an utterance depends on the previous discourse context, and the examples illustrate several ways in which this can occur. Topics are not only directly continued, but also interrupted for things such as clarifications or corrections. The interruptions themselves are also subject to similar interruptions. This chapter has also shown how various surface linguistic phenomena (for example, clue words) can be used to recognize particular relationships with the previous context as well as how the system can proceed without such clues. Thus, the implicit information regarding whether an utterance continues or interrupts a topic, as well as how, is made explicit. Finally, we have seen how the theory can be used to process consecutive utterances and indirect speech acts.

Chapter 5

Sentence Fragments, Ellipsis, and Plan Ellipsis

1. Introduction

This chapter continues the last by tracing through two more dialogues, this time concentrating on the processing of utterances with missing words or phrases, e.g. sentence fragments and elliptical utterances. The first section will be in the context of a new domain and will illustrate how the system, using the MODIFY-PLAN meta-plan, handles a type of ellipsis somewhat like the type typically processed by current natural language understanding systems. First, the example will be traced to show how much of the missing semantic information of the input can be recovered as a side effect of the plan recognition process. The same example will then be repeated, beginning the recognition procedure with the elliptical input assumed already filled in by some other means. As with other phenomena governed by a local discourse context, we will again see that while having the results makes the plan recognition task much easier, if such results are not available the plan recognizer will typically be able to reach the same interpretation, although by perhaps a more circuitous path.

The second example (another train dialogue) will illustrate the processing of sentence fragments as well as *plan ellipsis*, phrases that replace entities implied by rather than entities mentioned in a preceding utterance. As in the first example, the filling in of the ellipsis will be done via recognition of MODIFY-PLAN. Both examples will also reiterate many of the points

made in the last chapter. In particular, exactly the same plan recognition algorithm, stack manipulations, meta-plan schemas, and discourse analysis will be used for *all* the examples. Only the domain plan schemas ever change.

2. Ellipsis

This section simulates the system's processing of a version of Dialogue 3, simplified[1] as shown:

User: Could you mount a magtape for me? It's tape1.

Operator: We are not allowed to mount that magtape. You will have to talk to operator about it. After nine a.m. Monday through Friday.

User: How about tape tape2?

This example will show how the system, taking the role of the operator, understands the speaker's utterances, in particular the elliptical last utterance. Figure 5.1 shows the meta-plans that will be used in this example. The meta-plans were presented in Chapter 2, repeated without change in Chapter 4, and are again repeated without change for the convenience of the reader. Also, recall from Chapter 2 that the speech act decompositions shown in Figure 2.8 were supplemented with the following decompositions and associated constraints (the prerequisites, etc. of Figure 2.8 remain unchanged):

HEADER:	REQUEST(speaker, hearer, action)
DECOMPOSITION-5:	SURFACE-NP(speaker, hearer, noun-phrase)
CONSTRAINT:	CONTAINS (action, noun-phrase)

HEADER:	INFORM(speaker, hearer, proposition)
DECOMPOSITION-2:	SURFACE-NP(speaker, hearer, noun-phrase)
CONSTRAINT:	CONTAINS (proposition, noun-phrase)

The plan recognition algorithm begins with the parse of the user's first utterance:

SURFACE-REQUEST(user,system,
INFORMIF(system,user,CANDO(system,MOUNT(system,skolemMagtape)))).

[1]Recall from Chapter 1 that the user's initial turn consisted of two more utterances. With respect to this example, the utterances are superfluous and can be eliminated, enabling the example to focus on the processing of the user's next utterance.

HEADER:	INTRODUCE-PLAN(speaker, hearer, action, plan)
DECOMPOSITION:	REQUEST(speaker, hearer, action)
EFFECTS:	WANT(hearer, plan) NEXT(action, plan)
CONSTRAINTS:	STEP(action, plan) AGENT(action, hearer)
HEADER:	IDENTIFY-PARAMETER(speaker,hearer,parameter,action,plan)
DECOMPOSITION:	INFORMREF(speaker, hearer, term proposition)
EFFECTS:	NEXT(action, plan) KNOW-PARAMETER(hearer, parameter, action, plan)
CONSTRAINTS:	PARAMETER(parameter, action) STEP(action, plan) PARAMETER(parameter, proposition) PARAMETER(term, proposition) WANT(hearer, plan)
HEADER:	MODIFY-PLAN(speaker, hearer, change, changee, newAction, oldAction, oldPlan, newPlan)
PREREQUISITE:	WANT(hearer, oldPlan)
DECOMPOSITION:	REQUEST(speaker, hearer, newAction)
EFFECTS:	POP(CLOSURE(oldPlan)) NEXT(newAction)
CONSTRAINTS:	PARAMETER(oldAction, changee) STEP(oldAction, oldPlan) STEP(newAction, newPlan) EQUAL(newAction, SUBST(change, changee, oldAction)) EQUAL(TYPE(change), TYPE(changee)) ~EQUAL(change,changee) REPLACE(stack, oldStack)

Figure 5.1: The Meta-Plans Needed for the Tape Example (Repeated from Chapter 2)

Because the user has a particular tape in mind (as "It's tape1" later verifies), skolemMagtape will be used to refer to this as yet unknown constant of type magtape.[2] The name skolem-Magtape was chosen to reflect the similarity between unknown constants and first order predi-

[2]Since at this point in the dialogue the operator really doesn't know if the user has a particular tape in mind, technically another parse should be produced with "a magtape" represented as a variable. However, since any plans from such a parse will be eliminated with "It's tape1" the omission is not important.

cate calculus skolem functions [68], i.e. functions which explicitly define the dependence of an existential variable on universal variables. The actual implementation of items such as skolem-Magtape as skolem functions will be explained in Chapter 6.

Since at the beginning of a dialogue there is no stack, the system will try to recognize an introduction of some user domain plan. From the SURFACE-REQUEST the system can perform forward chaining to an indirect REQUEST for the MOUNT (recall the speech act definitions of Figure 2.8), and from that to an introduction of a plan containing the MOUNT. A stack is constructed from the recognized meta-plan and its object plan as shown in Figure 5.2.

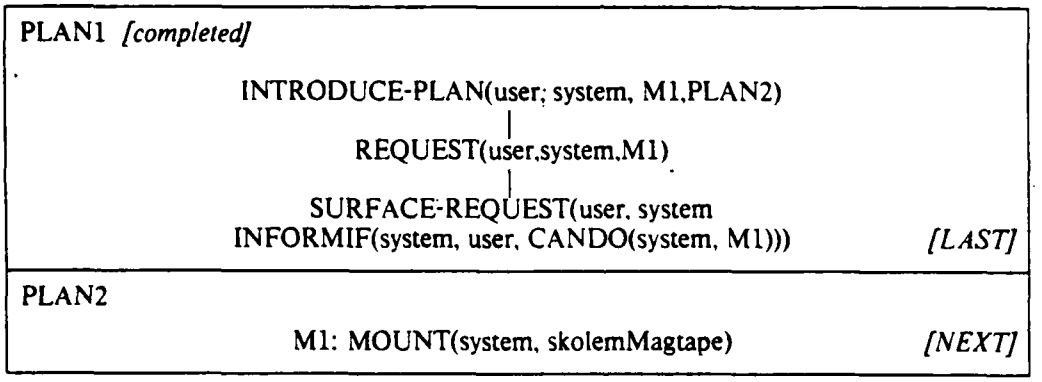


Figure 5.2: The Plan Stack After the First Utterance

The other possibility, chaining from the SURFACE-REQUEST (via decomposition (1)) to a literal REQUEST for INFORMIF, and from that to an introduction of a plan containing the INFORMIF, is eliminated by the heuristics since the effect of such an INFORMIF is already true, i.e. the user knows if the operator can mount tapes.

The parse of the user's next utterance, SURFACE-INFORM (user, system, EQUAL (skolemMagtape, tape1)) is then received (assuming the resolution of "it" by local means, as in the last chapter). From the SURFACE-INFORM chaining can proceed through INFORM to

an INFORMREF of skolemMagtape or of tape1, or through an INFORMIF. Chaining from the INFORMREF of skolemMagtape leads to recognition of an IDENTITY-PARAMETER of skolemMagtape in M1. The other possibilities, chaining from an INFORMREF of tape1, or from an INFORMIF, are eliminated. The former is eliminated via the heuristics, since saying tape1 is equal to an unknown constant of type tape provides no new information and thus the KNOWREF effect is already true; the latter is eliminated since it does not connect with any meta-plans. As in the last chapter, the INFORMREF is a preference (2) user elaboration of a stacked plan and is expected (since the plan introduced by the first utterance contained an unknown parameter, i.e. skolemMagtape), and will be preferred to any other meta-plans. The stack constructed after this phase of recognition is shown in Figure 5.3.

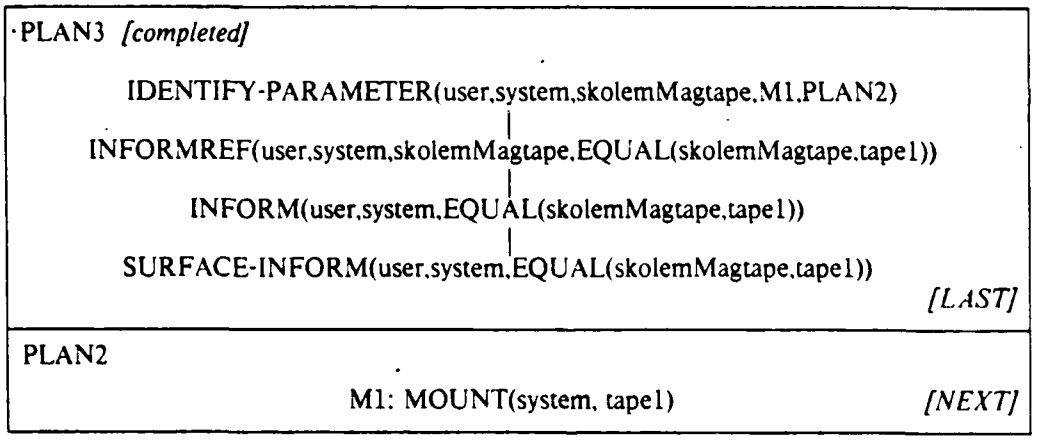


Figure 5.3: The Plan Stack after the User's Elaboration

Note how PLAN1 is first popped, in order to place the new meta-plan, PLAN3, above its object plan. While such a popping was expected since PLAN1 was believed completed, it was left on the stack until its popping was implicitly confirmed by recognition of PLAN3, just in case a clarification of the introduction, rather than the MOUNT, had unexpectedly followed. Finally, note the effects of the IDENTIFY-PARAMETER, binding skolemMagtape to tape1 in M1.

Simulating the system's response, the system examines the plan stack, pops the completed PLAN3, and resumes execution of M1, the next step in PLAN2. The system, however, is unable to actually perform the MOUNT and generates a response indicating this fact, as well as a suggested user debugging action.

The user reply "How about tape tape2?" is parsed as a clue word "how about" followed by a SURFACE-NP(user, system, tape2). As discussed in Chapter 2, various surface clues can be used to help decide whether such a fragment is part of an underlying REQUEST or an INFORM. For example, in this case, interrogative mood connects the SURFACE-NP to a REQUEST, and then using the appropriate decomposition we have

REQUEST(user, system, ?action)

where CONTAINS(?action, tape2)
AGENT(?action, system)

At this point, the clue word can be used to constrain the search procedure normally performed without the clue, i.e. looking for a continuation of PLAN2, then any meta-plan referring to PLAN2, is constrained to directly trying a MODIFY-PLAN of PLAN2. While "how about" can also be used to introduce a plan (e.g. "How about a bite to eat?), such an interpretation corresponds to a total topic change (the least preferred coherence heuristic) and is thus not yet pursued. The search yields the hypothesis MODIFY-PLAN (user, system, ?change, ?changee, ?action, ?oldAction, ?oldPlan, ?newPlan) where

(1) CONTAINS(?action, tape2)
(2) AGENT(?action, system)

Application of the heuristics then verifies that this plan is plausible. The system checks that the prerequisite, WANT(system, ?oldPlan), was true. This can be satisfied if ?oldPlan is bound to PLAN2 or PLAN3 in the stack in Figure 5.3. The constraints explicitly specify use of this old stack, since otherwise the user would be viewed as modifying the system's reply, rather than

the user's request.[3] PLAN2 is preferred via the coherence heuristics, since PLAN3 is completed and can be popped. Then, the system checks that there are no violations of the constraints, now instantiated to:

(3) PARAMETER(?oldAction,?changee)
(4) STEP(?oldAction,PLAN2)
(5) STEP(?action,?newPlan)
(6) EQUAL(?action,SUBST(?change,?changee,?oldAction)
(7) EQUAL(TYPE(?change),TYPE(?changee))
(8) ~EQUAL(?change,?changee)

Satisfaction of constraint (4) binds ?oldAction to M1, and satisfaction of (3) then binds ?changee to either system or tape1. However, to satisfy (6), i.e. to have ?action be a modification of M1, where either system or tape1 is modified, given constraints (1),(2) (the fact that system and tape2 are two parameters of ?action), (7) and (8) (the fact that the modifications are made by type compatible, but not equal, entities) only tape1 can be bound to ?changee. Finally, these bindings, along with satisfaction of constraint (5), results in the creation of a new plan, PLAN5, with step M2:MOUNT(system, tape2).

The effects of the recognized MODIFY-PLAN are asserted, in particular PLAN2 and any of its object plans (in this case, none) are popped from the stack, and M2 is marked as in focus. Finally, the recognized MODIFY meta-plan and its associated object plan PLAN5 are pushed on the (now empty) stack, as shown in Figure 5.4. Note how the unknown action in the user's original elliptical utterance has now been determined as a result of constraint satisfaction in the plan recognition process.

The interruption in this example is slightly different from those of the previous examples. In the last chapter we had topics that were interrupted but later resumed. Here we have topics that are interrupted but will not be resumed. Instead, the interrupted topic (in the above example, PLAN2) is replaced by a modified topic (PLAN5). Since this is a violation of the default

[3]Although it is not clear how many of the previous stacks need to be remembered, it is clear that some sort of history will have to be maintained. For example, Chapter 6 will show that plan recognizers are non-monotonic and can maintain previous contexts to retract incorrect assumptions.

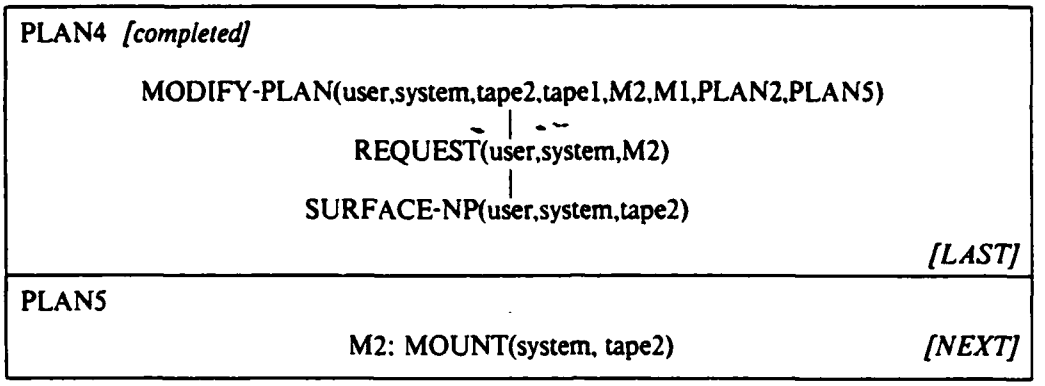


Figure 5.4: The Modified Plan Stack

stack behavior, one of the effects of the MODIFY-PLAN meta-plan is the explicit pop (and removal) of the interrupted topic.

As discussed in Chapter 3, the type of ellipsis illustrated in this dialogue is similar to that already handled by many existing systems in that the noun phrase "tape2" replaces the previous lexical item "tape1" (although it is not clear that systems based on a syntactic and semantic reformulation of the local context would actually be able to handle this utterance, due to the lack of a previous lexical item for the clue word "how about" to replace).

What would happen if a complete parse, i.e. SURFACE-REQUEST(user, system, M3:MOUNT(system, tape2)) was input to the plan recognition system instead of just the SURFACE-NP? From the SURFACE-REQUEST, one could directly chain to a literal REQUEST, and using the clue "how about" to MODIFY-PLAN(user, system, ?change, ?changee, M3, ?oldAction, ?oldPlan, ?newPlan). This is similar to the previous intermediate result, except that the binding of ?newAction to M3 is already known, and in fact the constraint satisfaction process will eventually result in the same bindings in both cases. However, the process this time will be much quicker since much of the work will involve verifying (rather than constructing interpretations to satisfy) the constraints. Thus, as with other

phenomena governed by local discourse context, plan recognition provides a back-up method of analysis if needed. Perhaps more importantly, however, the next section will show how exactly the same mechanism can handle a type of ellipsis not so similar to those normally covered by natural language systems.

3. Plan Ellipsis and Initial Sentence Fragments

This section simulates the system's processing of the beginning of the following train dialogue:

Passenger:	Trains going from here to Ottawa?
Clerk:	Ottawa. Next one is at four-thirty.
Passenger:	How about Wednesday?
Clerk:	One at nine thirty, nine thirty in the morning, four thirty in the afternoon...yeah, that's it.

We will see how the system's plan recognition process enables effective processing of the passenger's initial noun phrase, as well as how the MODIFY-PLAN meta-plan enables processing of a class of difficult elliptical utterances. The train domain plan schemas necessary for the example are repeated in Figure 5.5. The meta-plan schemas needed are the same as presented in Figure 5.1.

The parser's output for the first utterance is simply SURFACE-NP (person1, clerk1, departTrainSet1) where EQUAL (station(departTrainSet1), Ottawa). Furthermore, since the mood is interrogative and in this particular domain the clerk's role is to provide information via speech acts, we know this is very likely a REQUEST to perform either an INFORMIF or INFORMREF (as in Allen and Perrault [3]).

As with "The eight-fifty to Montreal?" of the last chapter, the passenger could be asking for some role value of the train set. The processing would be analogous to that of the earlier utterance, and an introduction of a clarification would be postulated. However, in this case no

HEADER:	GOTO(agent, location, time)
EFFECT:	AT(agent, location, time)
HEADER:	MEET(agent, arriveTrain)
DECOMPOSITION:	GOTO(agent, gate(arriveTrain), time(arriveTrain))
HEADER:	BOARD(agent, departTrain)
DECOMPOSITION:	GOTO(agent, gate(departTrain), time(departTrain)) GETON(agent, departTrain)
HEADER:	TAKE-TRAIN-TRIP(agent, departTrain, destination)
DECOMPOSITION:	SELECT-TRAIN(agent, departTrain, departTrainSet) BUY-TICKET(agent, clerk, ticket) BOARD(agent, departTrain)
CONSTRAINTS:	EQUAL(destination, station(departTrain)) EQUAL(destination, station(departTrainSet)) EQUAL(departTrain, object(ticket))

Figure 5.5: The Train Domain Plan Schemas (Repeated from Chapter 2)

domain plan has parameters corresponding to the unknown roles of departTrainSet1, i.e. to sets of gates or times, so the hypothesis ultimately fails. Alternatively, since in this dialogue we are inquiring about a train set rather than a single train, the passenger might instead be interested in knowing the elements of the set. From an introduction of this INFORMREF an identification of the last parameter in the SELECT-TRAIN plan can be postulated. The stack constructed from this interpretation is shown in Figure 5.6. Finally, no stack will be constructed from the INFORMIF, since no chain of meta-plans to a domain plan can be constructed.

Simulating the system, one possible explanation for the response could be as follows. The system pops the completed PLAN1 off the stack, and wants to resume PLAN2 by generating a response to achieve the INFORMREF I1. Suppose for some reason, the system decides

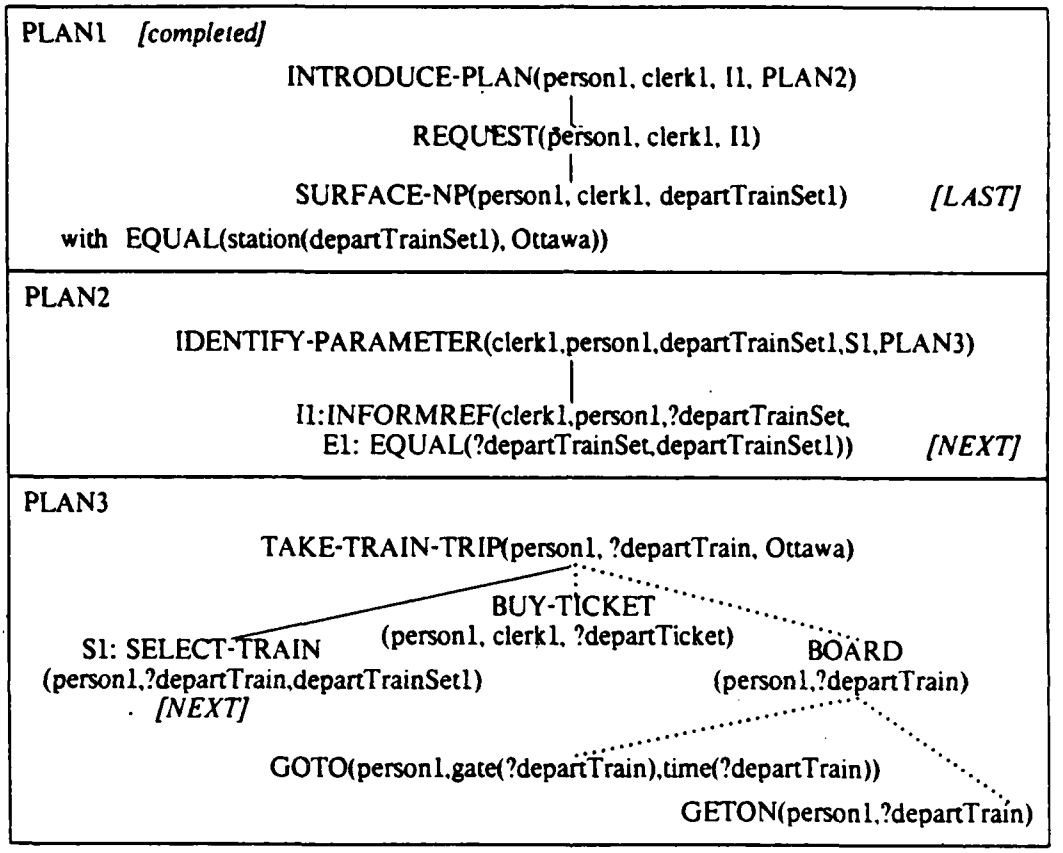


Figure 5.6: The Stack after the Initial Fragment

not to perform the intended response, but instead to perform a fancier one. Looking at PLAN1, the system realizes that the passenger's ultimate goal is to BOARD a specific train, and since the passenger is already in the train station, assumes that the next train is the most desirable. The system thus identifies only the next train, rather than all the trains, going to Ottawa.

Unfortunately, although the passenger is currently in the train station, the train to be boarded leaves on a later date. Thus, the passenger tries to acquire the needed information with "How about Wednesday?" This utterance is parsed as the clue word "How about" and a SURFACE-NP (person1, clerk1, Wednesday), which as above, is likely a REQUEST to

perform some type of system INFORM involving Wednesday. "How about" signals not only that an interruption has occurred, but also how the interruption may be related to the interrupted topic. In particular "how about" signals that either an INTRODUCE-PLAN or MODIFY-PLAN is probably being executed. Since the latter is preferred by the coherence heuristics that hypothesis is tried first, overruling the default expectation to pop the completed PLAN2 and resume PLAN3. In other words, if "how about" was not present, recognition of a continuation of PLAN3 would have been attempted first. Chaining to MODIFY-PLAN yields MODIFY-PLAN (person1, clerk1, ?change, ?changee, ?action, ?oldAction, ?oldPlan, ?newPlan) where (1) ?action is some sort of system INFORM involving Wednesday. The heuristics are then applied to check the various instantiations as candidate plans. Since the constraints indicate MODIFY-PLAN uses the context of the oldStack (here the stack of Figure 5.6 rather than the stack after the system's response), the prerequisite WANT(clerk1, ?oldPlan) can be satisfied by PLAN1, PLAN2 or PLAN3. Since PLAN2 is preferred via the coherence heuristics, PLAN1 is popped and the PLAN2 binding is tried first. The rest of the parameters are bound via satisfaction of the following constraints:

(2) PARAMETER(?oldAction, ?changee)
(3) STEP(?oldAction, PLAN2)
(4) STEP(?action, ?newPlan)
(5) EQUAL(?action, SUBST(?change, ?changee, ?oldAction))
(6) EQUAL(TYPE(?change), TYPE(?changee))
(7) ~EQUAL(?change,?changee))

Constraint (3) can be satisfied by binding ?oldAction to I1 or IDENTIFY-PARAMETER, but with constraints (1) and (5) we know it must be bound to I1. Then, with (1), (2), (5), (6), and (7) ?action gets further specified to an INFORMREF with departTrainSet2, where EQUAL(time(departTrainSet2), Wednesday). Finally, satisfaction of constraint (4) results in the creation of a new plan, PLAN4, containing the new INFORMREF.

Now that MODIFY-PLAN (person1, clerk1, E2: EQUAL (?departTrainSet, departTrainSet2), E1: EQUAL (?departTrainSet, departTrainSet1), I2: INFORMREF (clerk1, person1,

?departTrainSet, E2), I1: INFORMREF (clerk1, person1, ?departTrainSet, E1), PLAN2, PLAN4) has been instantiated to satisfy the heuristics, chaining can continue. MODIFY-PLAN is not a step in any other plans, but since it has introduced PLAN4 a recursive recognition is performed. As with the initial INFORMREF I1, from I2 the recognizer can chain to an IDENTIFY-PARAMETER of SELECT-TRAIN. PLAN5 is introduced to contain this SELECT-TRAIN, and another recursive recognition procedure chains from SELECT-TRAIN to the higher level plan to take a train. Finally, since TAKE-TRAIN-TRIP is a domain plan, the recursive recognition procedure halts. Note how once the modified step was found, the rest of the plan stack had to be re-recognized in order to propagate the modification. The various effects of all the plans can now be asserted. In particular, the effect of MODIFY-PLAN pops PLAN2 and its object plan PLAN3 off the stack.. The new MODIFY-PLAN, its object plan PLAN4, and PLAN4's object plan PLAN5, can then be pushed on the now empty stack, as shown in Figure 5.7.. This time the system responds with the INFORMREF explicitly requested; the fact that the set contains trains with times during Wednesday blocks the reasoning behind the last response (when the unspecified time range was given a default by the system).

4. Summary

This chapter extends the last chapter by tracing through two more dialogues, one of which is in yet another domain. While some of the meta-plans illustrated in the last chapter are used in these examples, this chapter concentrates on the use of the MODIFY-PLAN meta plan. We have seen how the clue word "how about" is used to explicitly mark not only that an unexpected interruption, but also what kind of interruption, has occurred. As with the other interrupting meta-plans, execution of MODIFY-PLAN suspends execution of its object plan. However, MODIFY-PLAN modifies the plan stack so that execution is resumed with a modification (and replacement) of the goals of the interrupted object plan and its chain of

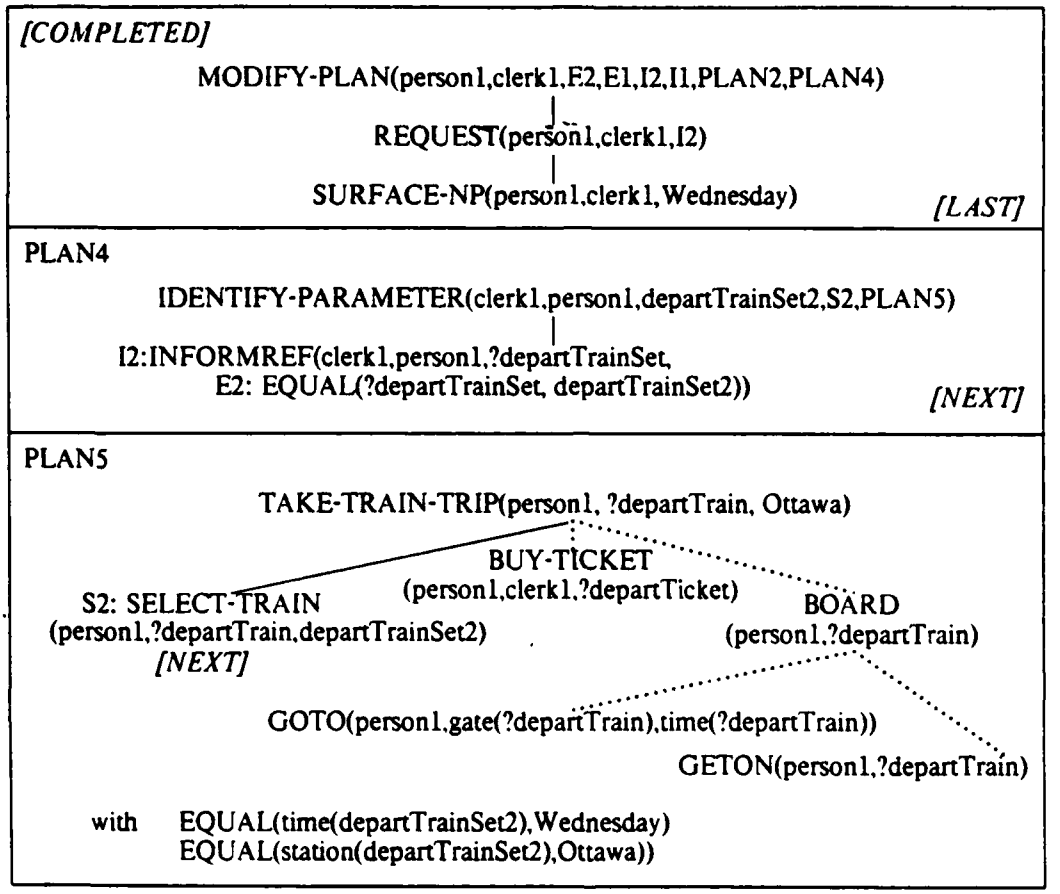


Figure 5.7: The Modified Plan Stack

object plans. This is in contrast to IDENTIFY-PARAMETER and CORRECT-PLAN, where further changes to the object plan are only constructive, i.e. new information may be present, but old information is left unchanged.

The type of processing captured by MODIFY-PLAN handles various types of elliptical processing as a special case. In particular, the reasoning implicit in the recognition of this meta-plan can explain intentionally the type of linguistic ellipsis typically processed using a local discourse context for reformulation. As with other linguistic phenomena dependent on a local discourse context, the plan recognizer can proceed with or without such results as input.

either using such linguistic input to speed the recognition process, or using the recognition process to explain the linguistic phenomena on an intentional basis, respectively. Furthermore, the same plan mechanism can be used to handle other, more difficult cases of elliptical utterances, for example omissions that cannot be filled using just the form of the preceding utterance.

Chapter 6

Knowledge Representation for Plan Recognition: Issues and their Implementation

1. Introduction

Any system that draws on a body of general intentional knowledge to explain observed behavior has fundamental requirements that tax the capabilities of existing knowledge representation systems. This chapter will discuss two such issues, both related to equality reasoning. The next section will show why the concept of unification must be extended to meet the requirements of the plan recognition task. A later section will show why a context dependent reasoning (and thus equality) system is needed. Each section will begin by illustrating why such a knowledge representation capability is necessary, and how such issues have been addressed by others (although unfortunately the issues raised are as yet largely unsolved). The solutions implemented in this particular system will then be presented, and illustrated via actual program transcripts. In general, the implemented solutions are straightforward rather than elegant, designed to meet the criteria of simply enabling the implementation given the available tools rather than making fundamental contributions in the area of knowledge representation.

To understand the transcripts, it will first be necessary to digress and explain HORNE terminology, notational conventions, and the various axioms used to represent the initial knowledge of the plan recognizer, i.e. the type hierarchies, plan schemas, and parser output. Section 3 will present this information, as well as detail the important constraint satisfaction mechanism of the plan recognition algorithm. (With the knowledge representation issues out of the way, the transcripts can finally be used to illustrate the implementation of the plan recognition algorithm.) The last section will then elaborate on the implementation's relation to the theory and simulations of the previous chapters.

2. Consistency Unification

2.1. The Problem

Consider the following true story:

> One day Diane suggests to Susan that Susan become the new editor of YUM-YUM, the departmental guide to eating out. Several weeks later Diane reads a message from Nemo, the current YUM-YUM editor, soliciting new reviews since a new editor will be taking over shortly. When Diane sees Susan again Diane comments on Susan's new role as editor. Unfortunately, Susan replies with bewilderment since Susan doesn't know anything about it.

The process Diane used to set "Susan" equal to "the new editor of YUM-YUM" is an example of consistency unification. While unification [68] typically results in the assertion of an equality between a constant (or a variable) and a type compatible variable, and, if extended with equality, between a constant and another constant if they can be proven equal, *consistency unification* allows an unknown constant such as "the new editor of YUM-YUM" to be set equal to a known constant (such as "Susan"), as long as the new equality is consistent with information already known (here, that Susan is a possible future editor of YUM-YUM). This example also shows that since such equality assertions are not logically sound, when faced with new information such equalities may later need to be retracted. This occurs when Diane finds out that Susan doesn't think she was the person Nemo was referring to.

In the context of the plan recognition system described in the previous chapters, the need for consistency unification arises at several points. The level of description given in the previous chapters glossed over one of the reasons such unification is needed. Recall the representation of a typical plan schema, as repeated in Figure 6.1.

HEADER:	INTRODUCE-PLAN(speaker, hearer, action, plan)
DECOMPOSITION:	REQUEST(speaker, hearer, action)
EFFECTS:	WANT(hearer, plan) NEXT(action, plan)
CONSTRAINTS:	STEP(action, plan) AGENT(action, hearer)

Figure 6.1: A Typical Plan Schema (Repeated from Chapter 2)

Since this schema applies to any INTRODUCE-PLAN abstract action (where the INTRODUCE-PLAN type has four roles), in the header *speaker, hearer, action,* and *plan* are actually variables. However, in the decomposition *speaker, hearer, action,* and *plan* are actually functions of the INTRODUCE-PLAN variable, since the decomposition means that any instantiation of the schema will be executed via a REQUEST, with the value of the first role of the REQUEST equal to the value of the first role of the INTRODUCE-PLAN, and so on. In other words, for all INTRODUCE-PLANS, there will exist an agent such that the agent is also the agent of the decomposition, and so on. The various roles of the REQUEST (and of the predicate in the effects and constraints) are thus analogous to first order predicate calculus *skolem functions*, functions that explicitly define the dependence of an existential variable on universals [68]. With this background we can now understand why the actual implementation of the schema looks more like that shown in Figure 6.2 (where ?I is a variable of type INTRODUCE-PLAN). Now we have come to the crux of the problem. As in many of the examples in the previous chapters, suppose a dialogue started with S1: REQUEST (user1,

HEADER:	?I:INTRODUCE-PLAN
DECOMPOSITION:	REQUEST(speaker(?I), hearer(?I), action(?I))
EFFECTS:	WANT(hearer(?I), plan(?I)) NEXT(action(?I), plan(?I))
CONSTRAINTS:	STEP(action(?I), plan(?I)) AGENT(action(?I), hearer(?I))

Figure 6.2: The Implementation of a Typical Plan Schema

system, action). The system/plan recognizer would query the HORNE reasoning system to find plan schemas that could have such a REQUEST as a decomposition. The INTRODUCE-PLAN schema of Figure 6.2 would provide one such candidate, but only if S1 could unify with the REQUEST of the decomposition. For this to happen, user1 must unify with speaker(?I), and so on, i.e. constants must unify with skolem functions. Unfortunately, unless these are already known to be equal, normal unification will not allow such a match to be made.

While at first glance it might appear that the problem is due to the particular implementation of this thesis, Charniak and McDermott [21] demonstrate that the need to match skolem functions in predefined schematic knowledge structures with constants in observations will occur in any similar task (such as story understanding) independently of the particular implementation chosen. This is true whether the skolem functions are literally skolem functions of first order predicate calculus or only conceptually so. Recognition of any instance of a schema implies the existence of a schema instantiation and of its roles. In predicate calculus terms we have existential variables and thus skolem functions, analogous to those described above.

A similar need to allow unification of skolems and constants occurs when skolems introduced by the parse are updated via the incorporation of new information. For example, recall Dialogue 2:

(1) User: Show me the generic concept called "employee."

(2) System:OK. <system displays network>

(3) User: I can't fit a new ic below it. Can you move it up?

(4) System:Yes. <system displays network>

(5) User: OK, now make an individual employee concept whose first name is "Sam"...

In utterance (3), the user is using "a new ic" referentially, i.e. the user has in mind a particular ic. The parser should have represented this entity as a skolem (which, as will be seen in the next section, will be a function of the plan context). Then, when processing utterance (5), in order to fit (5) into the existing plan context the skolem function has to be unified with a constant representing the concept Sam Jones.

Finally, when pronoun resolution is done intentionally (as opposed to basically linguistically), consistency unification will again be needed. Recall the discussion in Chapter 4 regarding the intentional resolution of "it" in "Can you move it up?" The skolem function for "it" is set equal to the constant representing the generic concept, in order to achieve constraint satisfaction during plan recognition. Charniak and McDermott [21] have similarly pointed out that representing pronouns with a skolem rather than a variable correctly captures the fact that the speaker has a particular entity in mind.

2.2. Solutions

2.2.1. Theoretical

Pople [74] was concerned with the mechanization of abduction, a more general version of the type of reasoning discussed above. Abduction is one of the three fundamental modes of logical reasoning tasks, along with deduction and induction. Abduction takes general rules and specific observations and hypothesizes an explanation for the observations. For example, given

(1) For all x, P(x) implies Q(x)
(2) P(a)
(3) Q(a)

abduction would take (1) and (3) and hypothesize (2). This is in contrast to deductive inference (e.g. from (1) and (2), (3)), and induction (e.g. from (2) and (3), (1)). Although deduction underlies most current knowledge representation work, it does not support the type of synthetic reasoning needed for many "intelligent" problem solving activities, in Pople's case medical diagnosis. While induction does allow the synthesis of hypotheses, such hypotheses are limited to generalizations. To allow abduction, Pople embedded deduction in an interactive hypothesize and test procedure guided by the principle of Occam's razor. In particular, abduction allowed the assumption of additional axioms that then enabled typical deductive processes to succeed. In the much more specific case of consistency unification, abductive reasoning would thus correspond to the assumption of equality axioms between constants and skolem functions.

As discussed in the last section, Charniak and McDermott [21] are also concerned with tasks requiring the matching of constants and skolem functions. They argue that performing what they call *abductive matching*, unifying two things if they are non-monotonically equal, enables exactly the reasoning needed for plan recognition and story comprehension as well as for pronoun resolution. In turn, a skolem function and constant are *non-monotonically equal* if asserting the equality is *consistent* (in the sense used with respect to default logics (Reiter [78])) with everything else in the knowledge base. Thus unification can occur between skolem functions and constants even if the necessary equality relation cannot be proved.

2.2.2. Practical

While the preceding idea is conceptually simple, it presupposes a system capable of proving consistency. Since such a capability is not a part of HORNE (or of most systems), for the purposes of this thesis an alternative mechanism had to built on top of the existing HORNE typed unification facilities. This section will describe what was done, using the transcript of the processing of "The eight-fifty to Montreal" as an illustration.

Recall that at the beginning of the dialogue the plan recognizer tries to chain from the input to a higher level goal by matching the input with a step in a decomposition in one of the expected plan schemas. For example, assuming an initial utterance such as

R1:REQUEST(person2, system, action7)

a query such as

(prove (DECOMPOSITION ?plan R1))

would be made with respect to the plan schema library. As discussed above, such a match will not succeed since skolem functions of the plan schemas cannot unify with constants of the input. Using the INTRODUCE-PLAN schema of Figure 6.2 we can see that speaker(?I) cannot unify with person2, hearer(?I) cannot unify with system, and action(?I) cannot unify with action7. Thus since we cannot eliminate the skolem functions of the plan schema, in order to find a relevant plan schema from the input a modified query has to be made. Since only variables can unify with skolems (assuming type consistency), the only query that will retrieve the appropriate plan schemas is one with all the constants in the input replaced by appropriately typed variables. In our example, the above query would be replaced by

(prove (DECOMPOSITION ?plan ?r:REQUEST))

which would return

(DECOMPOSITION ?plan:INTRODUCE-PLAN
REQUEST(speaker(?plan), hearer(?plan), action(?plan)))

Note that it was the restriction on the introduced variable (here via typing knowledge) that made such a mechanism useful, i.e. without any restriction on ?r every plan would have been retrieved. Once retrieved, an instantiation of the schema can then be created with all skolem functions set equal to the appropriate constants replaced in the query. Thus a specific instance, say I5, of type INTRODUCE-PLAN is asserted into the type hierarchy, with

EQUAL(speaker(I5), person2)

EQUAL(hearer(I5), system)
EQUAL(action(I5), action7).

At this point such assertions are consistent, since all that is known about each skolem function is its equality with the typed variables of the plan header. If later a contradiction arises due to any such equality (i.e. if a constraint, prerequisite, or effect of the plan schema involving a skolem function cannot pass the heuristic tests), the plan and the the equality are discarded. For example, if when satisfying the second constraint of I5 (recall Figure 6.2) we find that the system is not also the agent of action7, the plan I5 will be eliminated.

Thus, at the end of the plan recognition process a plan instantiation will have been constructed, partly by unifying skolems and constants when such equality assertions respect all the information contained in the plan schema and the parse. The important point to note is that in finding plan instantiations to explain observed input, equality assertions are made between skolem function and constants. However, only plans recognized via assertions that are consistent with typing information, the schematic definitions, and the parser information will survive. (As mentioned above, such equality assertions are non-monotonic and may need to be retracted. In a later section we will see how using a context-dependent equality system replaces the need for retractions).

The next section will illustrate this algorithm using the transcript of the processing of "The eight-fifty to Montreal." While probably not of interest to the casual reader, the section will illustrate details of the implementation (and will also show the direct correspondence between the implementation and the earlier high level traces).

3. The Implementation

3.1. The Axioms

Recall that the implementation uses the HORNE reasoning system [6] and thus stores its knowledge as horn clause axioms. As discussed earlier, the system starts its recognition task

with a predefined type hierarchy and a library of domain and meta-plan schemas. Figure 6.3 shows a fragment of the type hierarchy.

```
------------------------------------------------------

(ISUBTYPE T#Human T#Anything)

(define-subtype T#Action T#Anything
      (R#agent T#Human)
      (R#object T#Anything))

(define-functional-subtype T#SurfaceRequest T#Action
      (R#object T#Action)
      (R#hearer-SR T#Human))

(define-subtype T#MetaAction T#Action
      (R#hearer-MA T#Human)
      (R#plan T#Action))

(define-functional-subtype T#Ask T#MetaAction)

------------------------------------------------------
```

Figure 6.3: Part of the System's Type Hierarchy

The type hierarchy is built using primitive HORNE predicates and is similar to semantic networks. For example, the predicate *ISUBTYPE* asserts that the first argument is an (immediate) subtype of the second. By convention, all types begin with "T#", and the predefined type *T# Anything* is at the top of every hierarchy. The predicate *define-subtype* is used to define complex types, e.g types with various roles. Its syntax is

```
(define-subtype T# subtype T# supertype
      (R# rolename-1 T# roletype-1)
      ...
      (R# rolename-n T# roletype-n))
```

where by convention all rolenames are prefixed with "R#." Thus, *T# Action* is a complex subtype of *T# Anything*, with two roles corresponding to the agent and object of the action. (Perhaps more intuitively, *T# Action* corresponds to transitive verbs). *T# SurfaceRequest* is one subtype of *T# Action;* besides inheriting the two roles of *T# Action*, the type of the second role is further constrained, and a third role is added. *Define-functional-subtype* is similar to

define-subtype, except that if the roles of any two objects are equal the objects are also equal.[1] *T# MetaAction* is also a subtype of *T# Action;* besides inheriting the two roles of *T# Action*, instances of *T# MetaAction* also have two more roles corresponding to the hearer (recall all meta-plans are done by speech acts), and the object plan. *T# MetaAction* in turn has subtype *T# Ask*.

As in semantic nets there are also predicates that assert that a particular constant is of a given type (these will be illustrated below). For complex types the syntactic variant

(C#T# type rolevalue-1 ... rolevalue-n)

will often be used to refer to such a constant, making its role information explicit. The "C#" stands for constructor function. Such constructor functions display a complex instance as the name of the type, followed by the fillers of the roles (as ordered in the type's definition). For example, (C#T# SurfaceRequest person2 action7 system) would be equal to the constant R1, where R1 is SURFACE-REQUEST(person2, action7, system) using the more informal notation of previous chapters. Particular role values of such instantiations can be accessed using the function "(f# rolename constant)." For example, (f# agent R1) will be equal to person2.

Figure 6.4 presents the meta-plan schema ASK, first at the level of description used in the earlier chapters, then as implemented as a HORNE axiom. (ASK is a simplified version of IDENTIFY-PARAMETER with the fact that it is INTRODUCED by a SURFACE-REQUEST, ignoring issues of indirect speech acts, compiled in). All axioms asserting plan schemas are of the form:

(planType (header (prerequisites) (decomposition) (effects) (constraints))).

i.e. a planType followed by a plan structure consisting of its name and lists of prerequisites, steps, effects and constraints. Thus, the type of the ASK plan schema is a meta-plan. The

[1] As currently implemented, the axiom is technically incorrect. This is because without a role for time the system would be unable to distinguish between different instances of SURFACE-REQUEST involving the same speakers and

```
HEADER:          ASK (agent, term, agent2, plan)
DECOMPOSITION:   SURFACE-REQUEST (
                                  agent
                                  INFORMREF(agent2, term, agent, plan),
                                  agent2)
                 INFORMREF(agent2, term, agent, plan)
EFFECT:          KNOWREF(agent, term, plan)
CONSTRAINT:      PARAMETER(plan, term)
```

```
(metaPlan  ( ?a:T#Ask
             nil
             (  (C#T#SurfaceRequest  (f#agent ?a:T#Ask)
                                     (C#T#Informref  (f#hearer-MA ?a:T#Ask)
                                                     (f#object ?a:T#Ask)
                                                     (f#agent ?a:T#Ask)
                                                     (f#plan ?a T#Ask))
                                     (f#hearer-MA ?a:T#Ask))
                (C#T#Informref  (f#hearer-MA ?a:T#Ask)
                                (f#object ?a:T#Ask)
                                (f#agent ?a:T#Ask)
                                (f#plan ?a T#Ask)))
             (  (knowref (f#agent ?a:T#Ask) (f#object ?a:T#Ask) (f#plan ?a:T#Ask)))
             (  (parameter (f#plan ?a:T#Ask) (f#object ?a:T#Ask)))))
```

Figure 6.4: The Meta-Plan Schema ASK and its Implementation

header of the schema is *?a:T#Ask* where *?a* is a variable of type *T#Ask*. This schema thus holds for all instances of ASK, which is what a schema should say. The axiom also indicates that while there are no prerequisites, there is a decomposition (consisting of two steps implicitly time ordered), one effect, and one constraint. The first step of the decomposition is a SURFACE-REQUEST, with its agent role filled by the value of the agent role of the ASK, its action role filled by an INFORMREF defined in terms of the role values of the ASK, and its hearer role filled by the filler of the hearer role of the ASK. In other words, the various roles of the SURFACE-REQUEST (and recursively, the roles of its role fillers) have been filled by

utterances.

skolem functions.

```
---------------------------------------------------------------------
More Type Hierarchy Axioms:

(ISUBTYPE T#Time T#Anything)
(ISUBTYPE T#Place T#Anything)
(ISUBTYE T#Thing T#Anything)
(ISUBTYPE T#Location T#Place)
(ISUBTYPE T#City T#Place)
(define-functional-subtype T#Train T#Thing
      (R#departLocation T#Location)
      (R#departTime T#Time)
      (R#departStation T#City)
      (R#arriveLocation T#Location)
      (R#arriveTime T#Time)
      (R#arriveStation T#City))
(define-functional-subtype T#GoTo T#Action
      (R#object T#Location) (R#time T#Time))
(define-functional-subtype T#Informref T#MetaAction)
(ITYPE eightFifty T#Time)
(ITYPE Montreal T#City)
(ITYPE person1 T#Human)       (ITYPE clerk1 T#Human)
---------------------------------------------------------------------
Axioms Representing the Parse of "The eight-fifty to Montreal:"

(define-instance (f#informref1 context) T#Informref
      (R#agent clerk1)
      (R#object (f#SK0001 context))
      (R#hearer-MA person1)
      (R#plan (f#SK0003 context)))
(ITYPE (f#SK0003 context) T#Action)
(define-instance (f#surfaceRequest1 context) T#SurfaceRequest
      (R#agent person1)
      (R#object (f#informref1 context))
      (R#hearer-SR clerk1))
(ITYPE (f#train1 context) T#Train)
(ROLE (f#train1 context) (R#SK0002 context) (f#SK0001 context))
(ROLE (f#train1 context) R#departTime eightFifty)
(ROLE (f#train1 context) R#arriveStation Montreal)
---------------------------------------------------------------------
Another Plan Axiom

(domainPlan (?g:T#GoTo
  nil
  nil
  ((at (f#agent ?g:T#GoTo) (f#object ?g:T#GoTo) (f#time ?g:T#GoTo)))
  nil))
---------------------------------------------------------------------
```

Figure 6.5: Other Axioms Needed for the Example

Figure 6.5 presents the rest of the axioms needed to understand the example transcript. First, some more type axioms are given to supplement those of Figure 6.3. (As mentioned above, to make the implementation a bit simpler some of the types are slightly different than described earlier. Conceptually, any such differences are irrelevant). The axioms representing the parse of "The eight-fifty to Montreal" are of more interest. (Recall that while an implemented parser producing these axioms does exist, it is currently not hooked up to the plan recognition system). In particular, the second define-instance asserts that an instance of type *T#SurfaceRequest* (defined in Figure 6.3) was observed, with *person1* filling the agent role, *(f#informref1 context)* as the action requested, and *clerk1* filling the hearer role, where *(f#informref1 context)* is a specific instance of type *T#Informref* as defined by the first define-instance. (The various functions such as *(f#informref1 context)* represent skolem functions of the current plan context and will be fully explained in the next section.) Two of the role fillers of the INFORMREF are not specified in the utterance and are filled in by skolem functions (thus making the INFORMREF and embedding SURFACE-REQUEST skolem functions). Skolem functions are used instead of variables because it is assumed that the speaker has particular role fillers in mind (as when skolems are used instead of variables for pronouns and indefinite noun phrases used referentially). *(f#SKO001 context)*, the object of description asked for, is some role value of train1 (recall the discussion of the parse in Chapter 4). This fact is captured by the first role assertion, where

(ROLE object rolename rolevalue)

is a HORNE axiom that asserts that *rolevalue* fills the role *rolename* of *object*. Hence, the first ROLE assertion states that the object of the INFORMEF, *(f#SKO001 context)*, fills some role *(R#SKO002 context)* of *(f#train1 context)*, where the information known about train is asserted via the last two ROLE assertions. Finally, *(f#SKO003 context)* refers to the object plan, which at this point is also unknown except for its type (known via the definition of *T#Informref*, a subtype of type *T#MetaAction)*. The last axiom encodes the primitive action

GoTo as a plan schema (with only an effect explicitly listed).

3.2. Meta-Planning Requirements

Ignoring consistency unification and context-dependent equality reasoning, the implementation of the plan recognition algorithm is also noteworthy for its representation and use of meta-plans. For example, recall that to allow such plans about plans, a vocabulary for referring to and describing plans is needed. Figure 6.6 presents some of the axioms used to implement a preliminary vocabulary, based on the representation of plan schemas described above (i.e. (planType (header (prerequisites) (decomposition) (effects) (constraints)))). Thus, the first two axioms state that something is a plan if it is either a domainPlan or metaPlan. For example, (plan ?p) can be proved by proving (metaPlan ?p), which can be proved by unification with a meta-plan schema axiom, for example the ASK axiom of Figure 6.4. To talk about the structure of plans we have the predicates defined by the rest of the axioms. Thus, ?term is a parameter of ?action if

```
-----------------------------------------------------------------
(plan ?p) < (domainPlan ?p)
(plan ?p) < (metaPlan ?p)

(parameter ?action ?term) <
           (plan (?action . ?rest))
           (ROLE ?action ?name ?term)
           (BOUND ?action)
(domainParameter ?action ?term) <
           (domainPlan (action . ?rest))
           (ROLE ?action ?name ?term)
           (BOUND ?action)

(step ?action ?step) <
           (plan (?action ?prerequisites ?body . ?rest))
           (MEMBER ?step ?body)

(constraint ?action ?constraint) <
      (plan (?action ?prerequisites ?body ?effects ?constraints))
      (MEMBER ?constraint ?constraints))
-----------------------------------------------------------------
```

Figure 6.6: Sample of the Vocabulary Supporting Meta-Plans

1) ?action is the header of a plan schema
2) the role ?name of ?action is filled by ?term
3) ?action must be bound to something other than a variable.

(The last condition is because we want our predicates to be satisfied by either existing or constructed plan instantiations). For example, suppose we have an axiom for a metaPlan with header A1, where A1 is equal to (C#T#Ask person1 term1 system plan1). Then,

(prove (parameter ?action person1))

succeeds with

(parameter A1 person1).

Similarly, ?step is a step of ?action if

1) ?action is the header of a plan
2) ?step is a member of the list of steps in the plan's decomposition.

The constraint axiom is analogous. Thus,

(prove (step A1 ?step))

succeeds with both

```
(step A1 (C#T#SurfaceRequest
                    person1
                    (C#T#Informref system term person1 plan1)
                    system))
(step A1 (C#T#Informref system term person1 plan1))
```

The constraints of a meta-plan use this vocabulary to specify explicitly how the meta-plan and any object plan must be related. By specifying such conditions as constraints, the process of constraint satisfaction can be used to either verify a meta-plan / object-plan relationship with an existing plan, or to construct a new object plan satisfying the relationships. For example, suppose that chaining from "When does the train leave?" leads to the recognition of the clarification meta-plan

(C#T#Ask user time(train) system ?objectPlan).

Recall that before pursuing this hypothesis, the system has to first make sure that the plan's constraints can be satisfied (using the constraint axiom of Figure 6.6 to access the constraint of the meta-plan, as defined in Figure 6.4), i.e.

(prove (parameter ?objectPlan time(train))).

In other words, the system must make sure that at least one existing or expected plan has a parameter that could be filled in by the time of a train. Using the GOTO domain plan schema (recall Figure 6.5), one proof of the query would be

(parameter (C#T#GoTo ?agent ?location time(train)) time(train)).

In other words, the first subgoal of the parameter proof retrieved a variable of type GoTo (the header of the GoTo plan schema). The second goal made sure that some role of this action could have time(train) as a filler. The third goal then guaranteed that an instantiation of the variable (which satisfied the preceding but no other requirements) would be returned by the query. Thus, from a recognized meta-plan, general knowledge about likely plans, and requirements that any object plan would have to meet, the system can introduce an object plan specified only as necessary to satisfy the constraint. Furthermore, if there were previous utterances and the system had already expected a specific GoTo, say (C#GoTo user location1 time1), the query would also return

(parameter (C#GoTo user location1 time1) time1),

assuming the consistent unification of the skolem function time(train1) and the constant time1. In this case the system finds an existing plan that satisfies the constraints, which is the interpretation preferred by the coherence heuristics. In both cases, however, note that from a single utterance the system not only recognizes a plan but also explicitly relates it to a context of other plans.

3.3. The Transcript

With the above background, the transcript illustrating the implementation of consistency unification, and of course of the plan recognition algorithm, can now be presented. The comments have been inserted for the purposes of this section and appear in italics. A few formatting changes have also been made for greater readability.

```
Script started on Tue Dec  4 18:45:16 1984
$ savedlisp

 ...
```

Load reasoning system, plan recognition algorithm, axioms for type hierarchy, plan library, and parser output

```
7.(converse)

User utterance: (f#surfaceRequest1 context)
```

The utterance parse is input to the plan recognizer (currently by me)

```
Recognizing from:

Tree with root (f#surfaceRequest1 context)
```

Chaining begins from the observed utterance act (which if an abstract action is also the root of a tree of subactions)

```
((C#T#SurfaceRequest person1 (f#informref1 context) clerk1))
```

The constructor function printout of (f#surfaceRequest1 context)

```
Current context is (f#surfaceRequest1 context)
```

Contexts will be explained in the next section

```
Checking
(step ?planAct:T#Action (f#surfaceRequest1 context))
```

Chaining begins by trying to find any plan (with ?planAct as header) that contains the observation as a step (where step is an axiom appropriately defined on the plan data structure)

```
Querying:
(step ?planAct:T#Action (f#surfaceRequest1 context))
```

First, the query is tried with regular unification. This is because once there is a stack, the coherence heuristics will try to first match with existing plan instantiations, which do not necessarily contain skolem functions

```
Querying:
(step ?planAct:T#Action ?y00123:T#SurfaceRequest)
```

Since at the beginning of a dialogue matching is done with plan schemas, constants in the input could not be unified with skolem functions in the schemas. The query is adjusted by replacing all constants (including skolem functions) with appropriately typed variables. Now the skolem functions in the schemas can be matched with the query.

```
Query Result is:
(step ?planAct:T#Ask
      (C#T#SurfaceRequest
                (f#agent ?planAct:T#Ask)
                (C#T#Informref (f#hearer-MA ?planAct:T#Ask)
                               (f#object ?planAct:T#Ask)
                               (f#agent ?planAct:T#Ask)
                               (f#plan ?planAct:T#Ask))
                (f#hearer-MA ?planAct:T#Ask)))
```

The plan schema with an action of type T#Ask as a header is found. Any SurfaceRequest which is its step must be a skolem function of the Ask, as shown. In other words, the agent role of any SurfaceRequest which is a step of an Ask plan is filled by the agent of the Ask...

```
Asserting: (ITYPE (f#y00124 context) T#Ask)
```

Now that the schematic knowledge suggests a candidate plan, an instantiation of such a plan is created (and believed to be the plan being executed by the speaker)

```
Current Context is (f#y00124 context)

Asserting:
(EQ (C#T#SurfaceRequest
        (f#agent (f#y00124 context))
        (C#T#Informref (f#hearer-MA (f#y00124 context))
                       (f#object (f#y00124 context))
                       (f#agent (f#y00124 context))
                       (f#plan (f#y00124 context)))
        (f#hearer-MA (f#y00124 context)))
    (f#surfaceRequest1 context))
```

The step of the candidate plan is "unified" with the step observed. If the constant and skolem are already equal, fine. If not, they are asserted equal.

```
Asserting:
(EQ (f#agent (f#y00124 context))
    (f#agent (f#surfaceRequest1 context)))
```

Since the step is a complex type, every role of the candidate step is set equal to every role of the observed step (and so on recursively, if the role itself is also a complex type)

```
Asserting:
(EQ (C#T#Informref (f#hearer-MA (f#y00124 context))
                   (f#object (f#y00124 context))
                   (f#agent (f#y00124 context))
                   (f#plan (f#y00124 context)))
    (f#object (f#surfaceRequest1 context)))

Asserting:
(EQ (f#hearer-MA (f#y00124 context))
    (f#agent (f#object (f#surfaceRequest1 context))))

Asserting:
(EQ (f#object (f#y00124 context))
    (f#object (f#object (f#surfaceRequest1 context))))
```

The hearer of the candidate Informref (the agent of the ASK) can already be proved equal to the hearer of the observed Informef using the HORNE equality system. This is because in each Informref the hearer role is equal to the agent of the embedding SurfaceRequest, and the two agents were already set equal by the second equality assertion above. Thus the equality assertion would be redundant and does not appear.

```
Asserting:
(EQ (f#plan (f#y00124 context))
    (f#plan (f#object (f#surfaceRequest1 context))))
```

Abductive unification has now been partially simulated. Skolem functions in the query were set equal to constants in the input, consistent with the typing information. The heuristics now check the consistency of the equality assertions with the other information known about the skolems (i.e. whether the prerequisites, effects, and constraints involving these skolem functions are consistent).

```
Checking the constraints of:
(f#y00124 context)
```

To be plausible, the constraints of any postulated plan must be satisfied.

```
Checking
(domainParameter (f#plan (f#y00124 context))
                 (f#object (f#y00124 context)))
```

The constraint of the plan instantiation (recall the Ask schema of Figure 6.4). Parameter is further constrained to domainParameter, since desiring simplicity the recognizer first tries to find a domain plan as object plan. This is just another heuristic controlling the search.

```
Querying:
(domainParameter (f#plan (f#y00124 context))
                 (f#object (f#y00124 context)))
```

As above, the query is first made using regular unification (and fails).

```
Querying:
(domainParameter ?y00175:T#Action ?y00176)
```

As above, the query is thus adjusted to enable matching with the schemas. However, since the object of any Ask is equal to the object of the Informref of its SURFACE-REQUEST (recall Figure 6.4), which in this case is equal to (f#SKO001 context) (recall Figure 6.5), the query will be more involved.

```
and
((ROLE ?y00177:T#Train ?y00180:T#U ?y00176))
```

Substituting an appropriate variable for the skolem function (f#object (f#y00124 context)) does not capture all the information known, since it's also equal to (f#SKO001 context), which is a role value of train1 (recall the first role assertion of Figure 6.5). The query must thus be adjusted to include this information. Note how the added query also has to be modified by replacing constants and skolems with variables (and possibly recursively more queries).

```
and
(BOUND ?y00176)
```

Another heuristic guiding the search: prefer the most constrained match for a skolem function.

```
Query Result is:
(domainParameter (C#T#GoTo ?x000307:T#Human
                           (f#departLocation ?y00177:T#Train)
                           ?x000327:T#Time)
                 (f#departLocation ?y00177:T#Train))
```

As above, an appropriate plan schema for the object plan is found. An instantiation will be created and the appropriate equalities asserted. (This will be explained fully in the next section).

```
Asserting: (ITYPE (f#y00181 context) T#GoTo)

Current Context is (f#y00181 context)

Asserting: (EQ (f#y00181 context) (f#plan (f#y00124 context)))

Asserting: (EQ R#departLocation (R#SK0002 context))
 (q- 1)(RETRACT (skolem (R#SK0002 context) ?rest))
 (r- 1)(RETRACT (skolem (R#SK0002 context) ?rest))

Asserting:
(EQ (f#departLocation (f#train1 context))
    (f#object (f#y00124 context)))

Asserting:
(EQ (f#departLocation (f#train1 context))
    (f#object (f#y00181 context)))

Checking the effects-false heuristic on
(f#y00124 context)
```

Now that the constraints are satisfied, try another heuristic.

```
Checking
(knowref (f#agent (f#y00124 context))
         (f#object (f#y00124 context))
         (f#plan (f#y00124 context)))

Querying:
(knowref (f#agent (f#y00124 context))
         (f#object (f#y00124 context))
         (f#plan (f#y00124 context)))

Querying:
(knowref ?y00206:T#Human ?y00207 ?y00210:T#GoTo)

and
((ROLE ?y00208:T#Train ?y00209:T#U ?y00207))

and
(BOUND ?y00207)
```

Both the original and modified queries fail. The effect is thus false and the plan still reasonable. Since there are no more heuristics to check, resume chaining (now from the just recognized candidate plan)

```
Recognizing from:

Tree with root (f#y00124 context)
```

The candidate plan, followed by the constructor function notation of it (with its step) and its object plan

```
((C#T#Ask person1 (f#SK0001 context) clerk1 (f#y00181 context))
 ((C#T#SurfaceRequest person1 (f#informref1 context) clerk1)))
Tree with root (f#plan (f#y00124 context))
((C#T#GoTo (f#agent (f#y00181 context))
           (f#SK0001 context)
           (f#time (f#y00181 context))))
      ...
```

4. Context-Dependent Reasoning

4.1. The Problem

Recall the analysis of the KLONE-ED dialogue as discussed in Chapter 4. After the user's first utterance ("Show me the generic concept called 'employee'") the recognizer could not determine whether an ADD-DATA or EXAMINE plan was being INTRODUCED. A separate stack representing each hypothesis was thus created, each embodying its own set of assumptions. For example, in one stack the object plan of INTRODUCE-PLAN contained ADD-DATA, while in the other stack the object plan of INTRODUCE-PLAN contained EXAMINE. Consequently, in each stack the observed SURFACE-REQUEST is equal to the step of a different INTRODUCE-PLAN. The set of assertions maintained by a plan recognition system must thus vary with each stack postulated.

A *context-dependent reasoning system*, e.g. a system that can remember multiple sets of assertions at a time, is absolutely essential for any incremental plan recognizer since by definition such recognizers maintain multiple hypotheses. As shown above, the processing of "Show me the generic concept called 'employee'" requires two sets of assertions. Subsequent utterances will be processed with respect to each set (unless new information eliminates one of

the hypotheses).

A context-dependent system will also prove useful for certain types of non-incremental plan recognition as well as for other types of hypothetical reasoning. Recall that in a non-incremental algorithm a plan recognizer is still allowed to investigate several hypotheses, although ultimately a best one will be chosen. If a reasoning system could only remember one set of assertions at a time, every time a hypothesis was reconsidered its axioms would have to be reasserted (and any other set of axioms retracted). Such a process would be very inefficient if the recognizer needed to frequently switch between the hypotheses.

4.2. Solutions

4.2.1. Theoretical

Hendrix [45] expanded the expressive powers of typical representation networks by allowing networks to be partitioned. Among the capabilities such partitions provided was a natural way of representing disjunction, since the information encoded by each disjunct could be placed (and reasoned about) in a different partition. Such a system would therefore support multiple sets of hypotheses.

Barton [10] has designed a reasoning system which, besides being a design variant of earlier systems that record justifications for conclusions, also supports fast hypothetical reasoning by remembering the consequences of several assumption sets at once. As in the RUP system (Reasoning Utility Package (McAllester [61])), Barton's system (called XRUP) uses expression grammars (McAllester [62]) to compactly represent the consequences of equalities. Unlike RUP, XRUP makes this representation primary, i.e. all statements are treated as equalities. The grammatical formalism is then extended to provide the capabilities of RUP, as well as to allow fast hypothetical reasoning. To do the latter XRUP distinguishes between permanent facts and temporary assumptions, and allows sets of such assumptions to be associated with a

data structure called a context.

Since XRUP expresses all statements as equalities, the system is most suited for situations that have a lot of equalities to begin with. Similarly, since the consequences of several assumption sets are remembered rather than recomputed, the XRUP context mechanism is primarily useful for systems that switch repeatedly among a small number of assumption sets (although the tradeoff is a need for more memory, as well as a much slower addition time for new facts since a new fact now has to be asserted in every context).

Currently Allen is investigating how to add an equality context mechanism to HORNE [6]. He proposes building a hierarchical context tree allowing the inheritance (as in type hierarchies) of equalities from less specific to more specific contexts. The mechanism HORNE uses to implement equality reasoning is not grammar based, but instead manipulates disjoint sets of equivalence relations using a union-find algorithm [1]. Since a context tree structure also allows for disjoint sets the plan is to use an appropriately modified union-find algorithm. Thus, although implemented very differently such a context mechanism would also allow several sets of equalities to be remembered at once (and ultimately would be extended to all assertions).

Recall the analysis of "Show me the generic concept called 'employee'," where the stacks (i.e. PLAN1 above PLAN2, or PLAN1 above PLAN3) are shown again in Figure 6.7. Although the system realizes the user is INTRODUCING a plan involving D1, when querying the plan libraries to see what the higher level plan is, two solutions are ultimately found. The system thus needs to assert

EXCLUSIVE-OR (EQUAL(plan(I1),PLAN2), EQUAL(plan(I1),PLAN3)),

but HORNE does not allow disjunction. Making the equality context-dependent would solve this problem since a context could be used for each hypothesis. Consider a representation of the necessary equalities using the context tree shown in Figure 6.8, where CONTEXT-1 and

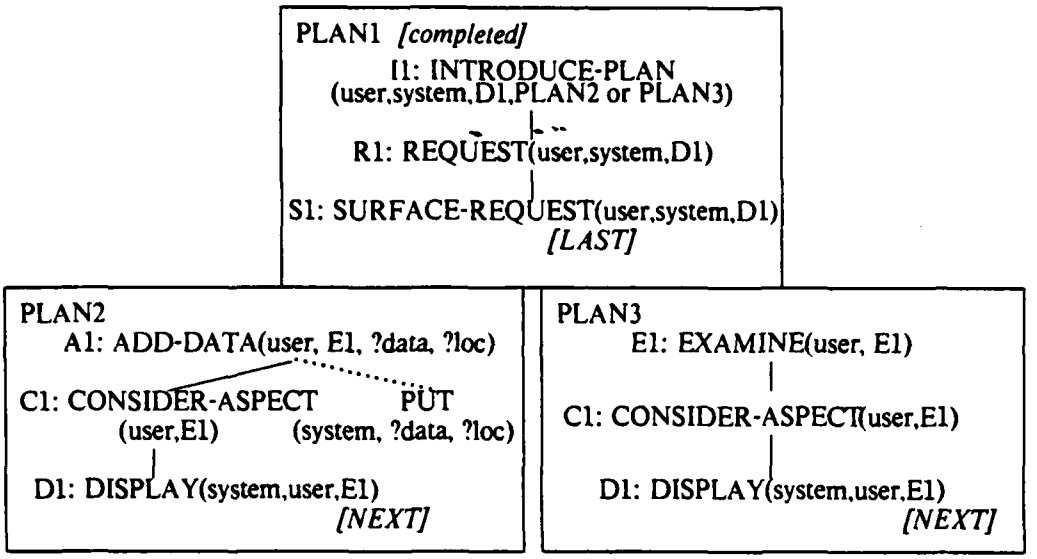


Figure 6.7: The Need for Multiple Sets of Assertions

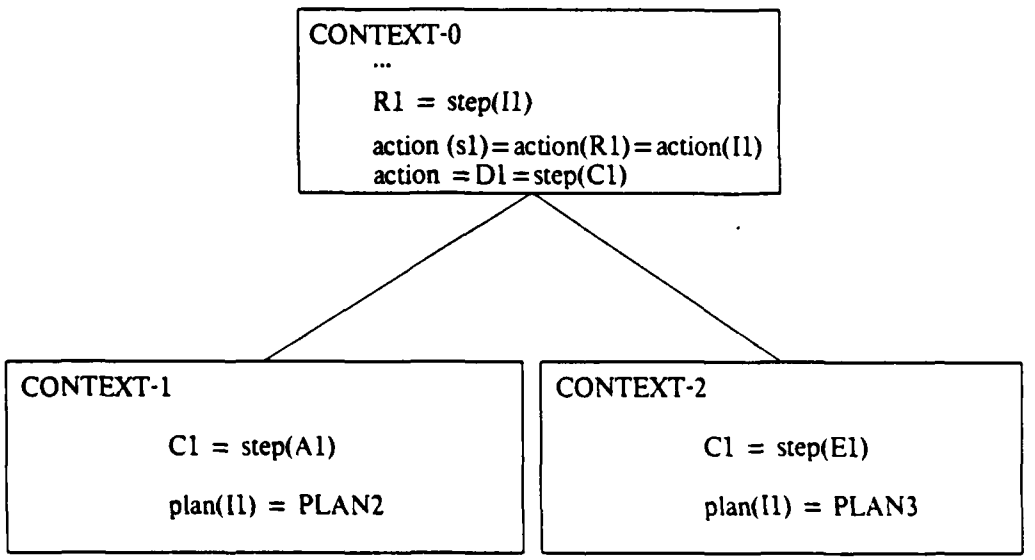


Figure 6.8: An Example Equality Context Tree with Inheritance

CONTEXT-2 consist of the equalities shown plus those inherited from CONTEXT-0.

4.2.2. Practical

Since contexts are currently not a part of HORNE such a mechanism had to be simulated. This section will describe what was done and illustrate the algorithm with the transcript of the processing of "The eight-fifty to Montreal."

In the plan recognition system described in this work, a new set of assertions will be needed every time a HORNE query has multiple proofs. (Subsequent steps of the plan recognition algorithm are then repeated for each result). Thus every time a query is made we want to allow a hook for the creation of new contexts, where a *context* is just a subset of assertions in the knowledge base. This can be done by creating a new context for every query result. The assertions made as a result of the query are added to the new context and the assertions from the parent context inherited. Thus, the plan recognizer will need to make assertions using a new predicate with an explicit context parameter. The obvious solution would seem to be a predicate that calls the HORNE assertion predicate, then associates the assertion with the specific context. While all assertions explicitly added to the knowledge base by the recognizer thus get indexed by a context, assertions internally added by HORNE do not (for example equivalence class axioms created adding a HORNE equality axiom). Instead the existence of the contexts must be compiled into the content of the assertions, for example by making any objects returned by a query a function of a context name.

For example, suppose the plan recognizer had partially recognized metaplan1 but still needed to find out what current plans would satisfy the requirements of the metaplan's object plan (as in the KLONE-ED dialogue). A query such as

```
(prove (OBJECT-PLAN ?plan metaplan1))
```

might be made, which returns the two answers

```
((OBJECT-PLAN PLAN2 metaplan1)
 (OBJECT-PLAN PLAN3 metaplan1)).
```

Assuming neither plan could be eliminated by the heuristics, the system would want to record both hypotheses, i.e.

(assert (EQUAL PLAN-ROLE(metaplan1) PLAN2) context2)
(assert (EQUAL PLAN-ROLE(metaplan1) PLAN3) context3).

Each assertion is now associated with a different context. Unfortunately, the HORNE equality system knows nothing about contexts and will create an equivalence class of (PLAN-ROLE(metaplan1), PLAN2, PLAN3), losing the relevant context information. To solve this problem, assertions such as the following are made instead:

(assert (EQUAL PLAN-ROLE(metaplan1) PLAN2(context2)))
(assert (EQUAL PLAN-ROLE(metaplan1) PLAN3(context3))).

From the point of view of HORNE this is still semantically incorrect since we now have the equivalence class (PLAN-ROLE(metaplan1) PLAN2(context2) PLAN3(context3)). However, the recognizer implementing such a context mechanism will know not to reason with assertions involving different contexts. Thus, the system would be able to use the information that PLAN-ROLE(metaplan1) equals PLAN2(context2) (if operating in context2) and PLAN-ROLE(metaplan1) equals PLAN3(context3) (if operating in context3), but not the information that PLAN2(context2) equals PLAN3(context3).

A side effect of this representation is that functions of contexts can also be used to represent unknown (i.e. skolem) constants in the parse, for example pronouns. Each possible resolution for the pronoun corresponds to an assumption made in a different hypothesis.

With respect to a few more details, as mentioned above a new context is a superset of the parent context and will explicitly list the inherited, as well as new, assertions. By allowing new assertions in a first query result to contain functions of the old context name, inherited assertions can just be copied. However for the context of subsequent disjunctive results, functions of a new context name will be needed and thus all inherited assertions re-asserted with functions dependent on this new context. Finally, in the current implementation contexts are never

actually deleted (a costly retraction process of all assertions containing functions of the retracted context). Although such assertions are still in the knowledge base, since the recognizer will no longer pursue the context the assertions will be ignored if retrieved.

4.2.3. The Transcript

This section will reiterate the above points using another train transcript. To avoid contexts in the last section, the previous transcript had the search constrained so that only one query result was returned. In this transcript the queries will return two results. (Other transcripts with more query results exist but are not shown since they are more cumbersome to follow and do not illustrate any new points with respect to contexts). The trace will show how from one initial hypothesis the system will create four (two from the original and within each of these two more), although ultimately two will be eliminated by the heuristics.

```
Script started on Mon Dec 10 20:04:29 1984
$ savedlisp
```

...

Axioms and algorithms loaded.

```
6.(converse)

User utterance: (f#surfaceRequest1 context)
```

All skolem functions in the parse are functions of the context "context"

```
Recognizing from:
```

Chaining begins from the utterance

```
Tree with root (f#surfaceRequest1 context)
((C#T#SurfaceRequest person1 (f#informref1 context) clerk1))
Current context is (f#surfaceRequest1 context)
```

This statement means that the plan recognizer just created a context, reusing the name "context," associated with its initial hypothesis (the action observed by the parser). The new hypothesis contains pointers to all the assertions of its context, i.e. (f#surfaceRequest1 context) contains pointers to the seven parser axioms of Figure 6.5. Since by using the name of the parent the functions do not need to be rewritten, the pointers reuse the axioms of the parent context. Since no new assertions are added in this case, the new context contains only those inherited.

Chaining is performed (using consistency unification as described in the last section).

```
Checking
(step ?planAct:T#Action (f#surfaceRequest1 context))
```

```
Querying:
(step ?planAct:T#Action (f#surfaceRequest1 context))

Querying:
(step ?planAct:T#Action ?y00123:T#SurfaceRequest)

Query Result is:
(step ?planAct:T#Ask
      (C#T#SurfaceRequest
                  (f#agent ?planAct:T#Ask)
                  (C#T#Informref (f#hearer-MA ?planAct:T#Ask)
                                 (f#object?planAct:T#Ask)
                                 (f#agent ?planAct:T#Ask)
                                 (f#plan ?planAct:T#Ask))
                  (f#hearer-MA ?planAct:T#Ask)))
Asserting: (ITYPE (f#y00124 context) T#Ask)
```

Since this query could have multiple results, a new context will be created for each. As above, the first context is again called "context" and will be a superset of the parent. The ASK created from the retrieved schematic knowledge is then asserted into this new context (simulated by making the ASK a function of "context").

```
Current Context is (f#y00124 context)
```

Again, an explicit message indicating that a new context "context" has now been created (i.e. appropriate axioms inherited) for the current hypothesis, the ASK meta-plan (f#y00124 context):

All new assertions are added to "context"

```
Asserting:
(EQ (C#T#SurfaceRequest
                (f#agent (f#y00124 context))
                (C#T#Informref (f#hearer-MA (f#y00124 context))
                               (f#object (f#y00124 context))
                               (f#agent (f#y00124 context))
                               (f#plan (f#y00124 context)))
                (f#hearer-MA (f#y00124 context)))
    (f#surfaceRequest1 context))

Asserting:
(EQ (f#agent (f#y00124 context))
    (f#agent (f#surfaceRequest1 context)))

Asserting:
(EQ (C#T#Informref (f#hearer-MA (f#y00124 context))
                   (f#object (f#y00124 context))
                   (f#agent (f#y00124 context))
                   (f#plan (f#y00124 context)))
    (f#object (f#surfaceRequest1 context)))

Asserting:
(EQ (f#hearer-MA (f#y00124 context))
    (f#agent (f#object (f#surfaceRequest1 context))))
```

```
Asserting:
(EQ (f#object (f#y00124 context))
    (f#object (f#object (f#surfaceRequest1 context))))

Asserting:
(EQ (f#plan (f#y00124 context))
    (f#plan (f#object (f#surfaceRequest1 context))))
```

Only one solution for the query was found. The plan heuristics evaluate the hypothesis.

```
Checking the constraints of:
(f#y00124 context)

Checking
(domainParameter (f#plan (f#y00124 context))
                 (f#object (f#y00124 context)))

Querying:
(domainParameter (f#plan (f#y00124 context))
                 (f#object (f#y00124 context)))

Querying:
(domainParameter ?y00175:T#Action ?y00176)

and
((ROLE ?y00177:T#Train ?y00180:T#U ?y00176))

and
(BOUND ?y00176)

Query Result is:
(domainParameter (C#T#GoTo ?x000307:T#Human
                           (f#departLocation ?y00177:T#Train)
                           ?x000327:T#Time)
                 (f#departLocation ?y00177:T#Train))
```

Constraint satisfaction finds that the ASK postulated from the observed SURFACE-REQUEST could refer to the departLocation role of a domain plan GOTO

```
Asserting: (ITYPE (f#y00181 context) T#GoTo)

Current Context is (f#y00181 context)
```

As before, since the query could yield several hypothesis a new context gets created for each.

```
Asserting: (EQ (f#y00181 context) (f#plan (f#y00124 context)))

Asserting: (EQ R#departLocation (R#SK0002 context))
 (q- 1)(RETRACT (skolem (R#SK0002 context) ?rest))
 (r- 1)(RETRACT (skolem (R#SK0002 context) ?rest))


Asserting:
(EQ (f#departLocation (f#train1 context))
```

```
    (f#object (f#y00181 context)))

Query Result is:
(domainParameter (C#T#GoTo ?x000307:T#Human
                           (f#arriveLocation ?y00177:T#Train)
                           ?x000327:T#Time)
                 (f#arriveLocation ?y00177:T#Train))
```

A second way to satisfy the constraint - ASK about the arriveLocation role of a GOTO plan

```
Asserting: (ITYPE c00206 T#Context)
```

We now have a second query result, i.e. our first branch in the context tree. Since this context will not be a superset of all previous contexts (it is a sibling rather than descendant of the context for the first query result), a new context name is needed.

```
Asserting: (ITYPE (f#p00208 c00206) T#GoTo)
```

As above an instantiation of the returned plan schema is created, this time functional on the new context "c00206."

Since the new context has a different name from its parent we can no longer inherit the parent's assertions (expressed as functions of "context") verbatim. Instead we need to reassert all such assertions with respect to "c00206." Note that instead of sharing a parent's context, each child explicitly list the parent's assertions in its own.

```
Asserting: (ITYPE (f#y00124 c00206) T#Ask)
```

The previously recognized ASK meta-plan is postulated in this context.

The parser's axioms are re-asserted.

```
Asserting:
(ITYPE (f#SK0001 c00206) T#Anything)

Asserting:
(skolem
    (f#SK0001 c00206)
    (ROLE (f#train1 c00206) (R#SK0002 c00206) (f#SK0001 c00206)))

Asserting:
(define-instance (f#informref1 c00206) T#Informref
                        (R#agent clerk1)
                        (R#object (f#SK0001 c00206))
                        (R#hearer-MA person1)
                        (R#plan (f#SK0003 c00206)))

  ...

Asserting:
(ROLE (f#train1 c00206) R#arriveStation Montreal)
```

The axioms of the parent context (of the ASK) are re-asserted.

```
Asserting:
(EQ (C#T#SurfaceRequest
                (f#agent (f#y00124 c00206))
```

```
                (C#T#Informref (f#hearer-MA (f#y00124 c00206))
                               (f#object (f#y00124 c00206))
                               (f#agent (f#y00124 c00206))
                               (f#plan (f#y00124 c00206)))
                (f#hearer-MA (f#y00124 c00206)))
    (f#surfaceRequest1 c00206))

Asserting:
(EQ (f#agent (f#y00124 c00206))
    (f#agent (f#surfaceRequest1 c00206)))

 ...


Current context is (f#p00208 c00206)
```

Now that all the axioms are inherited (i.e. asserted into context "c00206"), we have created an appropriate context for the new hypothesis and the new axioms can be asserted.

```
Asserting: (EQ (f#p00208 c00206) (f#plan (f#y00124 c00206)))

Asserting: (EQ R#arriveLocation (R#SK0002 c00206))
 (q- 1)(RETRACT (skolem (R#SK0002 c00206) ?rest
 (r- 1)(RETRACT (skolem (R#SK0002 c00206) ?rest))

Asserting:
(EQ (f#arriveLocation (f#train1 c00206))
    (f#object (f#y00124 c00206)))

Asserting:
(EQ (f#arriveLocation (f#train1 c00206))
    (f#object (f#p00208 c00206)))
```

The query is complete and the recognizer now has two hypotheses (an ASK regarding an arriveLocation and an ASK regarding a departLocation). All subsequent steps of the plan recognition algorithm are repeated for each context.

```
Checking the effects-false heuristic on
(f#y00124 context)
```

The heuristics are applied to the first ASK.

```
Checking
(knowref (f#agent (f#y00124 context))
         (f#object (f#y00124 context))
         (f#plan (f#y00124 context)))

Querying:
(knowref (f#agent (f#y00124 context))
         (f#object (f#y00124 context))
         (f#plan (f#y00124 context)))

Querying:
```

```
(knowref ?y00259:T#Human ?y00260 ?y00263:T#GoTo)

and
((ROLE ?y00261:T#Train ?y00262:T#U ?y00260))

and
(BOUND ?y00260)
```

The query fails (i.e. the effect is false)

```
Checking the effects-false heuristic on
(f#y00124 c00206)
```

...

The heuristics are applied to the second ASK (i.e. in the second context). As above, the plan's effect will be false and the hypothesis still viable. Since chaining has reached a branch point we stop. However, since the hypotheses involve meta-plans we recursively recognize from each object plan (from each GOTO).

```
Recognizing from:
```

(From the object plan in the first context)

```
Tree with root (f#plan (f#y00124 context))
((C#T#GoTo (f#agent (f#y00181 context))
           (f#SK0001 context)
           (f#time (f#y00181 context))))
Current context is (f#y00181 context)
```

Recall (f#y00181 context) is equal to (f#plan (f#y00124 context))

```
Checking
(step ?planAct:T#Action (f#plan (f#y00124 context)))

Querying:
(step ?planAct:T#Action (f#plan (f#y00124 context)))

Querying:
(step ?planAct:T#Action ?y00271:T#GoTo)

Query Result is:
(step ?planAct:T#Board
      (C#T#GoTo (f#agent ?planAct:T#Board)
                (f#departLocation (f#object ?planAct:T#Board))
                (f#departTime (f#object ?planAct:T#Board))))
```

The GOTO could be a step of a BOARD plan

```
Asserting: (ITYPE (f#y00272 context) T#Board)

Current Context is (f#y00272 context)

Asserting:
(EQ (C#T#GoTo (f#agent (f#y00272 context))
              (f#departLocation (f#object (f#y00272 context)))
```

```
            (f#departTime (f#object (f#y00272 context))))
    (f#plan (f#y00124 context)))

 ...

Asserting: (EQ (f#train1 context) (f#object (f#y00272 context)))

Query Result is:
(step ?planAct:T#Meet
      (C#T#GoTo (f#agent ?planAct:T#Meet)
                (f#arriveLocation (f#object ?planAct:T#Meet))
                (f#arriveTime (f#object ?planAct:T#Meet))))
```

The GOTO could also be part of a MEET plan

```
Asserting: (ITYPE c00297 T#Context)
```

With respect to our first ASK meta-plan hypothesis and its GOTO, we now have two ways to chain in the object plan. Thus the parent context "context" splits into the children contexts "context" (for the previous query result) and "c00297" (for this query result).

```
Asserting: (ITYPE (f#p00299 c00297) T#Meet)
```

The second possible higher-level object plan is asserted into "c00297"

The assertions of the parent context are inherited, i.e. reasserted into the new context

```
Asserting: (ITYPE (f#y00181 c00297) T#GoTo)

Asserting: (ITYPE (f#y00124 c00297) T#Ask)

 ...

Current context is (f#p00299 c00297)
```

Now that c00297 has inherited the appropriate axioms, the axioms related to the MEET hypothesis can be asserted.

```
 ...
```

The query chaining from GOTO returns no more results. Heuristics now evaluate each result.

```
Checking the constraints of:
(f#y00272 context)
```

Heuristic processing proceeds in the first context. The heuristics verify the constraints of the object plan hypothesis BOARD.

```
Checking
(EQ (f#departStation (f#object (f#y00272 context))) Toronto)
```

The departStation of the train being boarded can be abductively made equal to Toronto.

```
 ...

Checking the constraints of:
(f#p00299 c00297)
```

The constraints of the second object plan hypothesis (i.e. of MEET) are checked.

```
Checking
(EQ (f#arriveStation (f#object (f#p00299 c00297))) Toronto)
```

The arriveStation of all trains being met is Toronto.

```
Querying:
(EQ (f#arriveStation (f#object (f#p00299 c00297))) Toronto)

Querying:
(EQ ?y00364:T#City ?y00365:T#City)

Query Result is:
(EQ ?y00365:T#City ?y00365:T#City)
Asserting: (ITYPE (f#y00366 c00297) T#City)

Asserting:
(EQ (f#y00366 c00297) (f#arriveStation (f#object (f#p00299 c00297))))

Assertion Fails: (EQ (f#y00366 c00297) Toronto)
```

The only way this can be proved is by asserting that the arriveStation (Montreal, from the parse) equals Toronto. Only constants that are really skolem functions can be abductively unified with other constants. Thus, the constraint cannot be satisfied and the context "c00297" no longer pursued.

```
|trapped - type "go" to continue|
| type "reset" to get to toplevel |
go
```

The next heuristic is applied in each context

```
Checking the effects-false heuristic on
(f#y00272 context)
```

The heuristic is tried in the BOARD context. (The BOARD axiom has no effect currently specified so this is trivially true).

```
Checking the effects-false heuristic on
nil
```

Since MEET has been eliminated, there are no more hypotheses to check. We have now concluded our first step of chaining from the GOTO object plan of our first ASK hypothesis.

We now initiate chaining from a different GOTO, the object plan of the second ASK hypothesis (context "c00206")

```
Recognizing from:

Tree with root (f#plan (f#y00124 c00206))
((C#T#GoTo (f#agent (f#p00208 c00206))
           (f#SK0001 c00206)
           (f#time (f#p00208 c00206))))
```

As above, chaining from the second GOTO will lead to appropriately related BOARD and MEET plans, with the latter again eliminated via constraint satisfaction.

...

The algorithm concludes with two hypothesis and creates a stack for each consisting of the meta-plan pushed on top of its object plan.

```
Stack: ((f#y00124 context) (f#plan (f#y00124 context)) . s00479)
Stack: ((f#y00124 c00206) (f#plan (f#y00124 c00206)) . s00480)
(((f#y00124 context) (f#plan (f#y00124 context)) . s00479)
 ((f#y00124 c00206) (f#plan (f#y00124 c00206)) . s00480))
7. (...


10.↑DBye
$ ↑D
script done on Mon Dec 10 22:26:13 1984
```

5. The Implementation and the Plan Recognition Algorithm

Although some of the details are different than in the analysis of "The eight-fifty to Montreal" given in Chapter 4, the implementation still illustrates the necessary points with respect to the plan recognition algorithm. For example, the implementation shows how the system performs forward chaining via HORNE queries. At the beginning of a dialogue the system has only schematic knowledge regarding expectations and thus all queries are made to the plan libraries. Because any plan recognizer initially must chain using schematic knowledge, an implementation of consistency unification could not be avoided. Furthermore, to consider multiple chains through the search space the context mechanism was also necessary. Each hypothesis constructed from the query is then evaluated by heuristic processing. In particular, the implemented heuristics eliminate plans with unsatisfiable constraints and already satisfied effects, and assert more equalities between constants and skolem functions. The implementation of constraint satisfaction was particularly important since it illustrated how at the beginning of a dialogue object plans are created from schematic knowledge via satisfaction of a meta-plan's constraints. The chaining process is repeated until the search produces either no more, or more than one, hypothesis. At this point, if any hypothesis is a meta-plan the incre-

mental recognition algorithm is recursively called on the object plan. Once all such recursive calls are completed a stack is constructed for each hypothesis.

Currently multiple sentences are not processed. With respect to the plan recognizer the only difference would be the implementation of more heuristics, i.e. the coherence heuristics that use the stack. With one sentence there is no previous stack and only the heuristic preferring the schemas needed to be implemented. Implementing the stack heuristics would just mean that more specific HORNE queries would be constructed, i.e. a query would specify whether the search is made with respect to the plan stack or the plan library and whether the plan being searched for can be type constrained (as when by the first coherence heuristic we want to search for only CONTINUE-PLAN meta-plans). Implementation of the discourse processing (i.e. focus of attention and clue words) would also involve only the formulation of a more constrained query. For example, if via the coherence heuristics the query were already constrained to search using the stack, the focus information encoded in the predicates LAST and NEXT suggests where in the stack to search. If a clue word is noted, the plan being searched for by the query would be type constrained to be the type of plan correlated with the clue word (a simple table look-up).

Finally, as noted above the structures used in the implementation are simpler than those used in previous chapters. For example, INTRODUCE-PLAN and IDENTIFY-PARAMETER were combined and implemented as ASK so that only one (as opposed to two) recursive recognition process was needed. However, even with this simplification the implementation still illustrates the interruption of a domain plan. Also, although the simplified (and less general) ASK involves the satisfaction of only one (rather than many) constraints, the point that constraint satisfaction creates object plans is still made.

There were several reasons for making such simplifications, as well as for using simulations rather than implementations of existing technology (as done with the parser). While in

theory HORNE was an ideal reasoning system, in practice HORNE was not. The role (i.e. complex typing) mechanisms of HORNE were still rather experimental and contained many bugs. Also, the reasoning system (particularly the equality reasoning) was fairly slow, and the implementation of the extra knowledge representation mechanisms added even more overhead. For example, a single plan recognition run from a single utterance, unintegrated with any other system components and limited to only single query results, took approximately one and a half hours on a Vax 750.

6. Summary

The purpose of this chapter was two-fold. At one level the desire was to show how work in plan recognition can influence work in knowledge representation, since any plan recognizer will need to perform types of reasoning not generally supported by current knowledge representation systems. In particular, consistency unification and context dependent equality reasoning were discussed and shown to be required by in any plan recognition system. At a more practical level, the chapter illustrated how such issues could be dealt with (i.e. implemented) using the existing capabilities of a typical reasoning system such as HORNE. Algorithms simulating such capabilities were described and illustrated via sample plan recognition transcripts from the train domain. Furthermore, with the knowledge representation aspects of the transcript explained, the plan recognition points illustrated by the implementation could be summarized. In particular, the implementation supported the claim that from a single utterance multiple plans and their relationships (i.e. meta-plans and associated object plans as related by the constraints) could be incrementally recognized and heuristically pruned.

The first part of the chapter showed how in the course of plan recognition objects that were represented as skolem functions would need to be set equal to constants. Three cases where skolem functions arose in the representation (schematic knowledge, pronouns, and referential uses of indefinite descriptions) were illustrated using the examples of the previous

chapters. A method of matching such skolem functions with constants in HORNE (similar to making non-monotonic equality assertions) was then presented, using standard unification but modifying the associated query and result.

The latter half of the chapter concentrated on context-dependent reasoning systems (including equality). Such systems are useful for tasks involving hypothetical as well as disjunctive assertions. A method for simulating the existence of multiple sets of assertions using a context tree was discussed.

Chapter 7

Comparisons to Related Work

1. Plan-Based Approaches to Natural Language Processing

1.1. The Early Work

1.1.1. Planning and Recognizing Speech Acts

Recall that Allen, Cohen, and Perrault developed a framework for the understanding [3] and generation [22] of single utterances based on the idea that acts of communication were planned in order to achieve specific goals. Their approach combined the insights of speech act theory (Austin [9], Grice [35], Searle [84]) (where utterances were viewed as actions achieving intended effects) with the computational formalisms of artificial intelligence work in robot problem solving. In other words, Allen, Cohen, and Perrault argued that linguistic actions (i.e. utterances) were planned in order to change the beliefs of a hearer just as physical actions (such as stacking blocks) were planned in order to change the state of the physical world.

For the purposes of this section, Allen and Perrault's work [3] on speech act recognition will be illustrative. Allen and Perrault showed how recognition of a speaker's goals in the domain of discourse could explain the generation of helpful responses (responses providing more information than requested) as well as the understanding of indirect speech acts and sentence fragments. Such linguistic behavior had proved problematic for previous approaches.

For example, Figure 7.1 presents their analysis of the following example of helpful behavior:

P: When does the Montreal train leave?
C: 8:50. Gate 7.

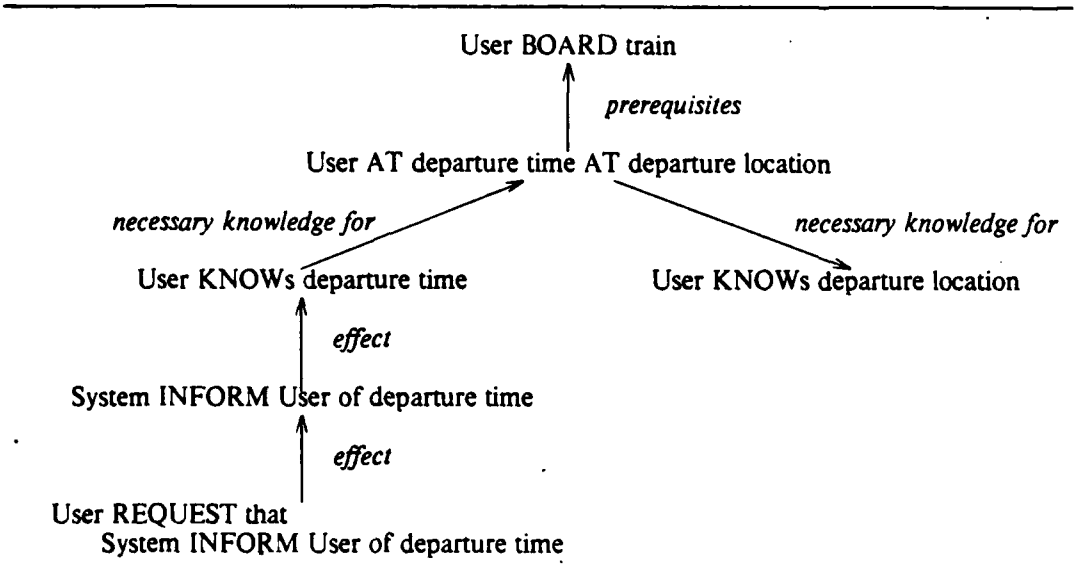


Figure 7.1: Plan Recognition and Helpful Behavior

From an initial speech act parse of the passenger's utterance (i.e. "User REQUEST that System INFORM User of departure time"), the clerk uses the representation for the speech act and domain plans, bottom-up plan representation based and knowledge based rules of inference, and top-down knowledge of likely domain goals (e.g. boarding or meeting trains) to infer that the speaker's underlying goal is to board the Montreal train. This plan is then examined for both explicit and implicit obstacles (here that the hearer believes that the speaker doesn't know the train's time or location, respectively), forming the basis for the clerk's response.

The major difference between Allen and Perrault's approach and the approach of this dissertation (excluding of course differences arising from understanding single utterances as opposed to dialogues) is that the new model has a hierarchy of plans, whereas all the actions in

Allen and Perrault are contained in a single plan. This has enabled the simplification of the notion of what a plan is and the solving of a puzzle that arose in the one-plan systems. Recall Figure 7.1; plans recognized in such systems were networks of action and state descriptions linked by causality and subpart relationships (i.e. prerequisite, effect, and decomposition relationships), plus a set of knowledge-based relationships. This latter set was not categorized as either a causal or a subpart relationship and so needed a special mechanism. Incorporating this into the action definitions would have required having a knowledge precondition for every term in every action. The problem was that these relationships were not part of any plan itself, but a relationship between plans. In the new system this relationship between plans is explicit, eliminating the need for the plan recognizer's special mechanism. For example, the "knowref," "know-pos" and "know-neg" relations of Allen and Perrault are modeled as constraints between a plan and a meta-plan, i.e., the plan to perform the task and the plan to obtain (i.e. clarify) the knowledge necessary to perform the task. In other words, the current work argues for the existence of not only domain intentions, but also higher level intentions indicating how all the intentions fit into the whole plan execution process. In more traditional communicative terms, the higher level intentions indicate how an utterance coheres with the previous discourse.

Regarding the analysis of indirect speech acts (ISA), in the present system a set of decompositions correspond to the conventional ISA. These are abstractions of inference paths that could be derived from first principles as in Allen and Perrault. Similar "compilation" of ISA can be found in Sidner and Israel [86] and Carberry [15]. It is not clear in those systems, however, whether the literal interpretation of such utterances could ever be recognized. In their systems, the ISA analysis is performed before the plan recognition phase. In the current system, the presence of "compiled" ISA allows indirect forms to be considered easily, but they are just one more option to the plan recognizer. The literal interpretation is still available and will be recognized in appropriate contexts. As far as the task is concerned, whether a request was

indirect or direct is irrelevant. For example, if we set up a plan to ask about someone's knowledge (say, by an initial utterance of "I need to know where the schedule is incomplete"), then the utterance "Do you know when the Windsor train leaves?" is interpreted literally as a yes/no question because that is the interpretation explicitly expected from the analysis of the initial utterance.

1.1.2. Understanding Discourse

Work in story understanding emphasized the importance of knowledge structures such as scripts [82], frames [66] and plans [82]. For example, Cullingford's [27] computer program SAM (Script Applier Mechanism) used scripts, stereotypical knowledge structures, to efficiently guide the understanding process. By providing extremely strong predictions the scripts greatly limited the drawing of inferences, although the tradeoff was that only stories corresponding to the stereotype could be understood. To understand novel stories, Wilensky [96] developed a story understanding program that reasoned about the situations of the stories in terms of the goals and plans of the characters. His computer program PAM (Plan Applier Mechanism) illustrated an algorithm for detecting and processing various types of goal-based stories, using knowledge about what goals existed, how goals were fulfilled, and how goals interacted. By adding bottom-up reasoning capabilities, his system could use small knowledge structures to construct script-like explanations. While Wilensky illustrated the importance of cognitive modeling as a way of efficiently inferring story coherence, he omitted most details regarding how this understanding process was actually done. Also, the work ignored most communicative issues. No connections were made between the underlying intentional structure and surface linguistic phenomena of the stories. Also, the work concentrated on recognition of the goals of the characters in the story, rather than on recognition of the goals of the "speaker" (i.e. of the author of the story). Finally, although Wilensky's stories illustrate many interesting goal interactions, goal suspension and resumption are not among them.

As discussed extensively in the previous chapters, Grosz [37] noted that since in task-oriented dialogues discourse structure followed the task structure, task structure could be used (along with surface linguistic phenomena) to construct an underlying discourse structure. Thus, for task-oriented dialogues containing subtask subdialogues the task structure performed a function similar to that of the scripts and plans in story understanders. Grosz, however, was primarily concerned with how the recognized discourse structure could then be used to account for linguistic phenomena. Explication of the recognition process was not of primary concern.

1.1.3. Plan Recognition for Other Tasks

The assumption that agents rationally achieve goals via plans has also proven to be useful outside the area of natural language understanding. For example, Genesereth [31] argued for the use of plan recognition in automatic consulting systems. He proposed a process in which a consultant would first reconstruct a user's plan (which the user believed to be correct), then analyze the plan for possible errors and thus the user's misconception. Plan recognition was responsible for the reconstruction phase and was viewed as the process of "parsing" a user's inputs, using a problem solving algorithm as grammar. A library of partial parses of both correct and incorrect plans could also be used to heuristically guide the process. Genesereth's plan recognizer thus had much in common with previously discussed efforts such as Allen and Perrault's [3], for example using plan-based rules of inference to search for a plan connecting goals and actions. However, there were several important differences as well. Allen and Perrault were concerned with not only recognizing, but also choosing between, alternative possible goals. In Geneserth's work the user's high level goal was already known. Furthermore, instead of pruning or rating various partial solutions during the recognition process, Genesereth simultaneously explored all possibilities until any solution was found.

A psychological concern with the process of plan recognition led Schmidt et al [83] to develop BELIEVER, a plan recognition system that inferred a structure of underlying inten-

tions from (a simple linguistic description of) a sequence of physical actions. Their theory was based on results of psychological experiments and encoded a hypothesize-and-revise algorithm. In particular, the plan recognition process consisted of setting up an expectation structure, matching observed actions against actions in the expectation structure, and revising the structure when its expectations are not matched. In turn, the revision process should recognize that revisions are needed, decide upon a set of revisions, and determine which revisions are effective. Since there are typically many ways to explain unexpected input, Schmidt et al. replaced an active control strategy choosing a revision with a simple wait and see strategy. This work in "story" understanding revolved around the desire to use and modify a single plan structure to explain variations of observed actions. The recognition algorithm of BELIEVER was particularly simple in its bottom-up processing. Recognition typically consisted of directly matching an observation and an expectation (or upon failure, a single revision). Unlike Allen and Perrault or the current work, there were no plan inference rules or heuristics controlling a much more complex search. Like PAM, BELIEVER performed keyhole recognition (as opposed to the intended recognition performed by the systems of Allen, Cohen, and Perrault). In other words, since the actions were not performed with the intent of enabling their recognition, assumptions based on the pragmatics of intended communication did not apply. For example, while BELIEVER's wait and see strategy was similar to the incremental control of dialogue systems, the former lacked the philosophical justification of the latter. Unlike SAM and PAM, BELIEVER was concerned with the accommodation of plan structures to the input, as opposed to the assimilation of the input to plan structures. This concern with plan revision overlaps somewhat with this work's concern with plan interruptions corresponding to corrections. However, as mentioned above the BELIEVER system's top-down "recognition" process does not address the issues of inference and search that characterize the current endeavor.

1.2. Discourse Extensions

Sidner [86, 90] outlined an approach that extended Allen and Perrault in many of the directions pursued in this work. For example, Sidner was concerned with understanding dialogues both intentionally and through the use of surface linguistic phenomena. She suggested a solution to a class of subdialogues that corresponded to debugging the plan in the task domain by allowing utterances to talk about, rather than always be a step in, task plans. Unfortunately, although her work allowed for multiple plans to be recognized from a given utterance such plans did not appear to be related in any systemic way. Also, her plan recognition system was never actually coordinated with the recognition of linguistic discourse markers.

The meta-planning based model of this dissertation grew out of many of Sidner's suggestions as well as earlier work at Rochester [4, 56], and can account for the debugging subdialogues Sidner discussed as well as other forms of clarification and correction. Since Chapter 4 presented an analysis of Dialogue 2, a minor variant of one of Sidner's dialogues, a comparison with Sidner's analysis will illustrate the advances of the meta-plan system. Consider the utterance " I can't fit a new ic below it." Upon receiving the parser's analysis of this utterance, Sidner's system uses a default rule that maps "I can't x" to "I want x." Basically this just retrieves the conventionalized meaning of the indirect speech act, which is then recognized as part of a plan structure in the normal way. However, there is another plan that is also triggered by the original utterance: if a user can't do something he or she wants, the user will execute a debug plan, temporarily suspending the blocked plan. Unfortunately, the details of how these two plans are related and how the suspension is managed are left unexplored. The meta-plan mechanism details the recognition of multiple plans and their relationships and provides a mechanism for managing topic suspensions and resumptions in the general case. For example, we saw the use of constraints to explicitly specify meta-plan/object plan relationships, and the use of constraint satisfaction to initiate recognition of an object plan from a meta-plan. Finally, although Sidner did not integrate her observations on discourse markers with her plan

recognizer, she did claim that interrupting relationships between multiple plans could not be recognized without such explicit discourse clues. There was one exception to this, which could be recognized using (previously unnecessary) conventions based on turn-taking. This is a very different claim from that made in this dissertation, where totally plan-based mechanisms for the recognition of multiple plans and their relationships were developed (and which could be coordinated with linguistic markers when available).

Carberry [15-17] was concerned with building a pragmatic component for a robust natural language interface. In particular, she developed TRACK, an incremental plan recognition system that could hypothesize and track task-level goals during information-seeking dialogues [15], then used this system to process pragmatically ill-formed input [16] and intersentential ellipsis [17]. TRACK extended the work of Allen and Perrault [3] by both processing dialogues rather than single utterances and manipulating more complex domain plans. In many ways TRACK was thus similar to the work of Sidner, ignoring issues of plan debugging. Carberry developed a set of heuristics based on principles for coherently shifting focus, since in information-seeking dialogues the underlying plan was not executed during the dialogue and focus shifts were thus not tightly constrained by the task structure. She then used these heuristics to update the context model after a new utterance. The context model was used to process utterances that violated pragmatic constraints by providing expectations useful in suggesting revisions to the query. The context model, supplemented with knowledge of discourse goals, was also used to process examples of intersentential ellipsis that could not be explained using the explicit information in the preceding utterance.

Carberry's tracking mechanism was solely intentional, i.e. it did not use any linguistic clues such as mode of reference or discourse markers to aid in its recognition task. Furthermore, information was always gathered with respect to a single plan, so there was no concern for recognition and suspension of plans. However, her emphasis on moving around in rather than following a task structure was unique and involved similar ideas of classes of coherency.

Her use of a plan context to handle violations of pragmatic constraints (i.e. pragmatic overshoot) was also novel. Unfortunately, for the problem of ellipsis resolution her theory proved inadequate and had to be supplemented with totally new knowledge and mechanisms regarding discourse goals that could be accomplished via elliptical fragments. For example, the goal to "Obtain-Corroboration" (elaboration and justification) would arise when the information-seeker was surprised; this goal was sometimes accomplished via an elliptical utterance. Her discourse goals were thus in the spirit of grammars of rhetorical predicates, although not enough details were provided to indicate if and how the problems with such approaches (as discussed in Chapter 3) were avoided. This is in contrast to the current work, where knowledge about a more general set of discourse goals in the form of meta-plans was incorporated into the theory from the beginning, enabling the use of a single framework to both track plans (including interruptions of plans) as well as process elliptical utterances.

Grosz [38], Levy [54], and Appelt [8] also advocated extension of the planning framework to integrate multiple perspectives, including both discourse and task goals. For example, Levy developed an initial formulation of communicative goals and strategies for processing narrative within a larger model involving planning of multiple goals.

> Some of these goals (called IDEATIONAL goals) are concerned directly with the communication of these ideas or propositions; some (called TEXTUAL goals) are concerned with the weaving of these ideas into a coherent text; and still others (called INTERPERSONAL goals) deal with presentation of self in relation to the hearer, with matters of status and attitude. [54]

(Levy's terms were borrowed from Halliday [42]). Appelt [8] concentrated on developing a planning formalism that could support the generation of utterances satisfying such multiple goals. For example, consider an utterance such as "Tighten the screw with the long philips screwdriver."

> [This utterance could] realize several illocutionary acts, like a REQUEST to tighten the screw and an INFORM that the tool for tightening the screw is the long philips screwdriver. Given that the speaker knows that the hearer doesn't know that a p rticular screwdriver is a philips screwdriver, the utterance could in that case also serve to inform the hearer that the long screwdriver is a philips screwdriver. This is contrasted with the case where "long" is used to distinguish long versus short. [8]

If formulated as an indirect speech act social goals such as politeness could also have been satisfied. Similarly, the choice of referring expressions and syntactic structure reflected goals regarding the shifting of focus. Although Appelt's work was notable for its formalism based on a possible-worlds semantics, as well as for integrating the planning of surface syntactic forms and speech act goals, he did not really address the issues of concern to the current work. For example, his planner did not generate extended discourse in any general sense. Multiple sentences were only generated when the planner could not satisfy its goals in one sentence. There were no global discourse goals; only immediate focus (Sidner [88]) was used to facilitate reference. Appelt himself points out the need to integrate results such as those of McKeown [64] and Reichman [76].

The ARGOT dialogue understanding system [4] included an effort to incorporate global discourse perspective into the plan-based framework of Allen and Perrault [3]. That early work has led to the development of much of what has been presented in this dissertation.

Grosz and Sidner [40] are currently integrating work on focusing in discourse and plan recognition. They argue that to adequately explain the discourse phenomena of clue words, referring expressions, and interruptions, discourse structure must explicitly be broken down into at least three distinct (but interacting) components: the structure of the sequence of utterances (a linguistic structure), the structure of intentions conveyed, and the attentional state at any given point. For example, Grosz and Sidner carefully differentiate between four classes of interruptions, showing how their theory can be used to explain aspects previously overlooked. Although the current work does not attempt a comprehensive investigation of interruptions, the theory of this dissertation seems to be consistent with many of Grosz and Sidner's ideas. For example, the three components of Grosz and Sidner's theory appear to have loose correlates in the current work's dialogues, set of recognized plans, and the stack of such plans (although Grosz and Sidner do not distinguish between intentions and meta-intentions). The matter of emphasis is quite different, however. Grosz and Sidner do not yet address the

process of actually recognizing their structures of intentions (e.g. identifying the plan algorithm and the information it uses), or show how the theory could be used to construct a discourse processing system.

Finally, the work of Goodman [32] demonstrated the use of multiple perspectives in a robust natural language understanding system. For example, [33] details a system that uses knowledge about language and the world to detect and repair by relaxation potential misunderstandings due to reference identification failure during task-oriented dialogues.

1.3. Planning Extensions

Cohen and Levesque [26] as well as Kautz [51] are interested in developing formal foundations for systems (such as the current one) that view language as planned behavior. For example, Cohen and Levesque use a theory of rational interaction, expressed in a logic based loosely on possible-worlds semantics, to derive the basis of a theory of communication. This is in contrast to the approach typically taken, where speech acts are considered theory primitives and thus need to be explicitly recognized. Kautz is concerned with formalizing the process of plan recognition by concentrating on the non-monotonic aspects of such reasoning. He develops a formal treatment of a simple case of (non-linguistic) plan recognition using the formal techniques of circumscription and minimal entailment. For example, he argues that a small set of general assumptions underlie the non-monotonic inferences performed in plan recognition. By characterizing these assumptions in terms of the minimization of certain predicates, Kautz is able to systematically augment the axioms of a theory of planning to yield just the implications desired for plan recognition. Plan recognition thus becomes the process of deriving a set of minimal models containing the original observations. Kautz is currently investigating how to extend his theory to recognize plans with more complex internal structure as well as higher-order plans such as meta-plans. While these rigorous formal treatments are very important, the trade-off is a return (at least for the near future) to the analysis of

extremely basic examples.

Pollack [73] is also concerned with improving the foundations of plan inference systems. She is concentrating on reformulating the underlying plan representations typically used in order to support more general forms of plan inference. For example, she notes that in questions to experts people often do not really know how to achieve their goals, and thus may refer to non-realizable plans. To enable the recognition of such plans, Pollack is formalizing a more expressive representation of plans based on philosophical accounts of action and intention. My work has implicitly assumed appropriate inputs and has thus not yet had to address these issues.

Many researchers in artificial intelligence have found it useful to incorporate meta-knowledge into their systems, although exactly what this term means as well as how such knowledge is used varies greatly from system to system (see, for example, the survey paper of Aiello [2]). Within the area of natural language processing, Woolf and McDonald [99], Carbonell [18] and Wilensky [97] have suggested the use of meta-knowledge. Woolf and McDonald are concerned with the generation of tutoring discourses that are grammar-based and context dependent. Meta-rules implement a context-dependent control strategy allowing digressions from a default rhetorical grammar. Carbonell noted that any comprehensive theory of discourse must address issues of meta-language communication as well as integrate the results with other discourse and domain knowledge, but did not outline a specific framework. This is in contrast to the computational model of the current work, which addresses many of these issues for an important class of dialogues. Wilensky has also shown how meta-planning knowledge (defined to mean general knowledge regarding a planner's goals, used during the construction process of any plan) can be used by plan-based natural language systems. For example, an instance of meta-planning knowledge is the fact that one way a planner can resolve a goal conflict is to abandon the less important goal. Such knowledge is necessary to explain observations that arise from the interaction of multiple goals. For example, meta-

planning knowledge could be used to explain the observation that John is listening to the news on the radio as John's resolution of his conflicting goals to buy a newspaper and to stay home. Making meta-planning knowledge explicit also enabled a more general resolution of some traditional planning problems, for example the use of special purpose mechanisms such as critics [80] to resolve goal interactions introduced during the process of hierarchical planning. This was because uniformly representing planning and meta-planning knowledge enabled use of the same planning and recognition processes for both goals and meta-goals. This was the strategy taken in the current work as well. Unfortunately, Wilensky's work suffered from the problems of his earlier work - while his taxonomies were excellent, the particular details of his theory were never well-specified. His work also differed from the current work in matter of emphasis. Wilensky's meta-plans were used to handle issues of concurrent goal interactions, while the meta-plans of this work were used to handle issues of hierarchies of goals.

2. Linguistic-Based Approaches to Discourse Processing

In addition to the above work, researchers have also addressed issues of discourse by developing computational models expressed solely in linguistic terms. For example, recall many of the results presented at length in Chapter 3. This section will briefly review and elaborate on the points made at that time.

Many surface linguistic phenomena could be explained in purely linguistic terms. Recall that viewing a discourse as a hierarchical structure of units of utterances, with units varying in the attention paid to them during the course of a dialogue, could explain phenomena such as a mode of reference and tense (Reichman [76]), use of clue words (Reichman [76], Polanyi and Scha [71], Cohen [25]), and so on. For example, choice of reference reflected which units of a discourse were currently being discussed, while clue words signaled shifts and relationships between the units. Typically linguistic (rhetorical) relationships such as "evidence," "support," "interrupt," etc. connected the units, as in the theories of Reichman [76], McKeown [64], and

Cohen [24]. As discussed above, Grosz's work [37] used the single relationship "subtask" and was thus an exception, although the tradeoff was analysis of only a limited class of task-oriented dialogues. The use of such rhetorical relationships was also supported by other criteria. For example, Hobbs [46] argued that such relationships were necessary in order to explain the underlying coherence of a discourse. Furthermore, he noted that as a side-effect of a hearer's recovery of these implicit relationships, issues of coreference were often resolved. Work in generation by McKeown [64] and Mann [60] showed how predefined structures of such relationships could be paired with discourse purpose, enabling effective (and coherent) organizations of text that did not just mirror the structure of the knowledge conveyed. Reichmans's grammar [76] for well-formed discourse and Woolf and McDonald's [99] rhetorical grammar for tutoring discourse reflected similar ideas.

While such concepts were intuitive they were extremely difficult to computationally formalize. Furthermore, the set of primitive relationships varied greatly form researcher to researcher and were thus a bit too subjective and intuitive. Most seriously, theories based on such predicates generally presupposed the computational recognition or generation of such predicates from linguistic input. For example, Cohen used an evidence oracle, Reichman [77] is waiting for the development of extremely sophisticated semantics modules, and Mann [60] notes that his theory is currently descriptive rather than constructive. McKeown's implemented system was a notable exception, although her predicates had associated semantics expressed in terms of the data base system and were thus not particularly general. These observations have led to the strategy of the current work, i.e. the reformulation of such notions in terms of (meta) planning and speech acts. This enabled a plan recognition algorithm to provide the link from the processing of actual input to recognition of underlying theoretical structures. It also enabled the earlier work on plan recognition to be combined with the associated surface phenomena results of the more linguistic works.

Linguistic analysis of a very local discourse context was also found to be useful for the explanation of surface phenomena. Sidner [88] (and more recently Grosz et al. [39]) showed how definite anaphora interpretation could be tractably performed using syntactic, semantic, and inferential knowledge. For example, syntactic constructions and grammatical relationships could be used to determine the discourse items that a speaker was centering on. These predictions could then be used to interpret anaphors in following sentences, interpretations that could be confirmed or rejected based on the presence of inferred contradictions. Similarly, many researchers showed how the syntactic and semantic information typical of most parsing systems could be used to understand a class of elliptical utterances (Grosz [37], Carbonell and Hayes [19], Hendrix [44], Waltz and Goodman [94], Weischedel and Sondheimer [95]).

While these results were very nice (and most importantly computationally tractable), the same phenomena could also be explained with extremely different kinds of theories. For example, the syntactic work of Sidner [88] and the rhetorical work of Hobbs [46] overlapped with respect to many issues of coreference. Dyer's work [29] illustrated that affect could even provide yet another perspective. Similarly, the elliptical utterances processed by the plan and discourse based approach of Carberry [17] formed a superset of those processed solely linguistically. Often such observations have resulted in statements that one theory should replace another. This is in contrast to the view that theories should try to incorporate these multiple perspectives (as discussed earlier and also as suggested as future directions in many of the linguistic works), explicitly acknowledging that linguistic behavior can and should be analyzed on many levels. For example, the current work has presented a system that coordinates complementary intentional, global linguistic, and local linguistic analyses. From a computational perspective, such an approach also enables the development of more robust, as well as tractable, systems.

3. Non-Computational Approaches

Many of the theories discussed above have both been influenced by and influenced non-computational works. As might be expected, many of the linguistics-based computational theories have some origins in work in linguistics. For example, Grimes [36] and Longacre [59] claimed that rhetorical predicates were organizing relations used at all levels of discourse. Within the area of linguistics there are also theories based on the idea that functional considerations must be taken into account when analyzing phenomena typically regarded as totally syntactic [43, 53, 69, 93]. For example, systemic linguistics (Halliday [43]) argues that since syntactic forms reflect meaningful choices of a speaker, a grammar should be developed that takes into account both functional and structural considerations. Some linguists have taken this functional view to the extreme and concluded that syntactic grammars do not even exist; phenomena previously regarded as syntactic are totally controlled by non-syntactic factors such as discourse function.

In the area of cognitive psychology, researchers such as Van Dijk and Kintsch [52] are concerned with developing theories of discourse processing that are compatible with more general theories of cognitive information processing. Many current computational theories seem intuitively consistent with general psychological results, for example segmentation, selective attention, use of expectations, and cues. Guindon [41] does a particularly noteworthy job in making such a connection. She constructed experiments illustrating relationships between psychological work on anaphora resolution (influenced heavily by the theories of Kintsch and Van Dijk) and computational linguistics work on focusing.

Work in ethnomethodology and conversational analysis (in the area of sociology) has shown that structural regularities occurring across many transcripts of spontaneous conversation can be explained by viewing conversation as an interactional, social process [48, 49]. In other words, when producing an utterance a speaker will make particular choices based on their

interactional consequences. The object of study in these paradigms is the methodology used by a speaker (hence the term ethno-methodology), rather than categories introduced by an analyst. In fact, structural regularities are both the formal conversational analyses and the tools used to achieve social regularity. For example, Sacks et al [81] show how a systemics for the organization of turn-taking can account for linguistic phenomena. In general, these works have emphasized sequential (as opposed to hierarchical) aspects of interactions, however.

Finally, as discussed above, plan-based approaches to language are greatly indebted to the work of Austin [9], Searle [84, 85], and Grice [35] in the philosophy of language.

Chapter 8

Conclusion

1. Summary

This dissertation has presented a computational theory and partial implementation of a discourse level model of dialogue understanding. The theory extends and integrates plan-based and linguistic-based approaches to language processing, arguing that such a synthesis is needed to computationally handle many discourse level phenomena present in naturally occurring dialogues, for example, interruptions, subdialogues, fragmental and elliptical utterances, and presence (as well as absence) of syntactic discourse clues. The simple, more tractable results of discourse analysis (for example, explanations of phenomena in terms of very local discourse contexts as well as correlations between syntactic devices and discourse function) have been left intact, while the more complex inferential processes relating utterances have been totally reformulated within a plan-based framework. Such an integration enables the handling of a wide range of linguistic behavior problematic for previous systems, while maintaining the computational advantages of the plan-based approach.

In order to handle a variety of subdialogues, including such interrupting subdialogues as clarifications and corrections, a new model of plan recognition was developed. In addition to the standard domain-dependent knowledge of task plans, domain-independent knowledge about the planning process itself was made available. This latter knowledge was formalized

using a set of meta-plans, and explicitly encoded execution relationships (or lack of such relationships) between plans and previous plans. A context dependent, incremental plan recognition algorithm was then developed, using the planning and meta-planning knowledge to recognize a hierarchy of domain and meta-plans underlying an utterance. In particular, the existing context was used to constrain initial postulation of (linguistically unmarked) meta-plans by preferring non-interrupting relationships to semantically related interruptions to totally unrelated interruptions. The formal constraints specifying the meta-plan / object plan relationships then enabled the use of constraint satisfaction to initiate recognition of an object plan from recognition of a meta-plan. The plans, along with the relationships between them, were then used to construct or update a stack representing the context of currently executing and suspended (i.e. interrupted) plans.

Many of these extensions to the earlier work in plan recognition can be seen as incorporating (and computationally formalizing) insights from linguistic-based explanations of communication. For example, implicitly relating subdialogues to one another in various rhetorically constrained ways was given an intentional correlate by incorporating meta-plans and context dependent preferences for their recognition. The observation that one of the possible rhetorical relationships was the (non) relationship of topic interruption led to the explicit recognition and manipulation of a stack of active and suspended plans. At the very least then, the theory can distinguish between interrupting and non-interrupting relationships.

In contrast, the more tractable linguistic results were left in totally non-intentional terms and instead coordinated with the process of plan recognition. For example, surface phenomena signaling otherwise implicit discourse relationships were made available at the start of plan recognition in order to overrule the default, totally inferential, intentional processing necessary in their absence. Similarly, linguistic resolutions of elliptical utterances and definite anaphora were used as inputs constraining the plan recognition process. Otherwise, these resolutions would have again been made totally intentionally via the more complex inferential

processes. Such interactions led to a system that was efficient as well as robust. By using syntactic results when available, the plan recognizer could improve its efficiency. However, the plan recognizer could also proceed totally intentionally when such linguistic clues were unavailable, and was thus much more robust than one dimensional systems.

The use of this integrated theory of dialogue understanding was demonstrated by detailing the processing of four example dialogues representing three domains. The examples illustrated the processing of many difficult discourse phenomena, for example interruptions corresponding to clarifications and corrections, interruptions of interruptions, resumption of interrupted topics, indirect speech acts, multi-sentential utterances, use of linguistic phenomena and analyses if available, and resolution of ellipsis, including plan ellipsis.

Finally, technical issues relating to the implementation were discussed. The implementation of the plan recognition algorithm for single utterances was presented, emphasising the incorporation of constraint-based meta-planning. The implementation of consistency unification and context-dependent reasoning, two modes of reasoning typically not supported by existing knowledge representation systems but necessary for any plan recognition task, was also illustrated.

2. Limitations and Future Directions

While this research has provided a framework for understanding an important class of dialogues, the scope of the work has been limited by a set of assumptions restricting both the theory and the data. Examination and relaxation of these assumptions yields several suggestions for future work. For example, the success of any plan-based theory crucially depends on the applicability of the assumption that people are rational agents who achieve a set of goals by generating and executing plans. Like other plan-based endeavors, this research has based its investigation on a set of dialogues for which this assumption is clearly true. In fact, recall that the Chinese cooking dialogues were deemed problematic due to the presence of interpersonal

exchanges unrelated to furthering goals relating to cooking. Thus, in order to demonstrate the true generality of a plan-based approach, work must be done on identifying and formalizing goals underlying less task-oriented conversations, for example the cooking dialogues and ultimately any informal, spontaneous conversations (such as found in Reichman's work [76]). Grosz and Sidner argue that such an endeavor will ultimately succeed. For example, with respect to their recent theory of discourse structure, they claim that

> One of the main generalizations of previous work will be to show that discourses generally are in some sense "task-oriented," but the kinds of "tasks" that can be achieved are quite varied -- some are physical, others mental, others linguistic. As a result, the term "task" is unfortunate and we will use the more general terminology of intentions -- speaking for example of discourse purposes -- for most of what we say. [40]

Wilensky's work makes a start on this endeavor by taxonomizing goals relevant to planning for everyday situations [97].

Even after limiting the analysis to task-oriented dialogues, the scope of this research was further constrained by analyzing only dialogues satisfying several simplifying assumptions. It was felt that the more complex issues could best be investigated within a well-understood framework for the simpler cases. For example, in all the domains the "system" had a joint goal with the user (or the goal of cooperating to achieve the user's goals). Investigation of dialogues in which the system has goals of its own would involve several extensions to the theoretical framework. Since the system's goals could potentially interact (both positively and negatively) with the user's goals, the system would have to be able to recognize user plans that resulted from these interactions. Incorporation of knowledge regarding goal interactions such as emphasized in Wilensky's work [97] would thus be necessary. Bruce [13] has also investigated interacting plans, particularly conflicting interactions that resulted in deception between agents. To support this type of interaction a more powerful belief model allowing non-shared beliefs would also need to be incorporated.

Another restriction on the dialogues analyzed was that they were (pragmatically) well-formed in a number of ways. In actuality speakers do make errors, although often these can

be recognized and repaired by the hearer. For example, consider the repair of the incorrect reference to a platform in the following dialogue:

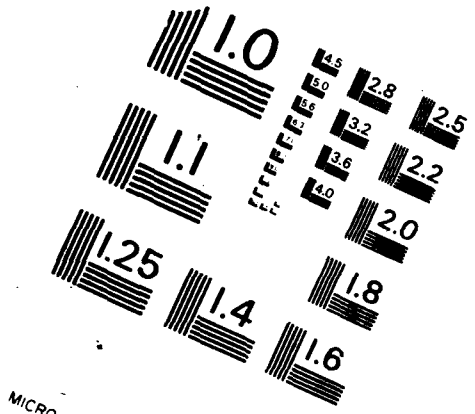
> Person: Could you tell me what platform the train from Cornwall comes in?
>
> Clerk: No platform at all. They, they all come in both sides downstairs here, and then they come up this ramp here.

Goodman [32] has shown how a system can use multiple sources of knowledge to recognize and then repair by relaxation reference identification failures. In a similar vein, Carberry [16] has shown how a system can use a plan context to revise queries that violate pragmatic constraints. In other words, when a system is unable to understand an utterance it should be able to use its violated expectations in order to identify and repair the user's miscommunication. General mechanisms for handling miscommunication thus need to be integrated with systems such as presented here. Furthermore, interactions between the two systems would need to be investigated. For example, a system must be able to decide when a reference failure indicates a topic switch as opposed to a miscommunication that needs to be repaired. Finally, miscommunication again points out the need for more sophisticated models of (non-shared) belief.

Besides incorporating other theoretical mechanisms to enable understanding of a wider range of dialogues, some further developments and refinements of the basic framework are also needed. For example, identification and formalization of a full taxonomy of meta-plans is needed. Also, recall that the analysis of the narrative aspects of the dialogues, e.g. of sequences of utterances that fill in a plan rather than execute or interrupt it, was fairly preliminary. A full treatment would require investigation of the many ways that a plan could be filled in (besides just identifying a plan's parameters as in the dialogues examined), as well as identification of both linguistic and non-linguistic clues signaling each type of filling in. A more complete treatment of surface phenomena is necessary as well. Recall that the current work concentrated on developing a framework that could incorporate linguistic results, rather than actually investigating the linguistic issues in their own right. Finally, a more sophisticated

AD-A170 871 PLAN RECOGNITION AND DISCOURSE ANALYSIS: AN INTEGRATED APPROACH FOR UNDERSTANDING DIALOGUES(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE D J LITMAN 1985 TR-170 3/3

UNCLASSIFIED N00014-82-K-0193 F/G 5/7 NL



incremental algorithm may be needed. Given that ambiguity is bound to occur in any reasonably complex domain (especially given the incomplete knowledge typical of most computational systems), cases where recognition should proceed beyond branch points (for example, to high level expectations as in Allen and Perrault [3]) need to be identified. On the other hand, although recognition of higher level domain plans provides a deeper level of understanding, not to mention understanding of otherwise problematic utterances, sometimes any plan recognition seems unnecessary. For example, recall that the domain plan analysis did not appear to be used by the system when responding to "Show me the generic concept called 'employee.'" Thus, further investigation on when to perform plan recognition, as well as when to stop it, is needed.

Investigation of how the framework developed in this dissertation could be used for tasks other than dialogue understanding is another interesting direction to pursue. For example, integration of work in discourse and goal analysis could usefully be applied to the largely unexplored area of dialogue dependent generation. (The current discourse work in generation has emphasized production of multi-sentential, single user utterances, i.e. it has concentrated on narrative rather than conversational aspects of discourse). McKeown et al [65] are making a start in this direction by showing how knowledge of a user's inferred goals can provide a perspective useful for tailoring a system's response. With respect to interruptions, work in generation typically ignores the issue of generation of interrupting subdialogues. Initiation of such subdialogues may be needed to enable generation of a response to an ambiguously recognized plan. Consider the following dialogue:

Person:	Going to Stratford, what gate would it be?
Clerk:	Which one is that?
Person:	Two-fifteen. I think is the ...
Clerk:	Yeah. Two-fifteen. Gate number eight.

In order to be able to provide a response, the clerk had to first generate a request for

clarification to disambiguate the user's plan and thus the user's question. In the following dialogue we see that the clerk needs to initiate several subdialogues in order to even achieve any sort of understanding.

Person:	Track eleven?
Clerk:	Track eleven.
Person:	Yeah.
Clerk:	Ah, you work around here?
Person:	No.
Clerk:	What do you want to go to track eleven for?
Person:	There's an employment office there. CP.
Clerk:	CP employment is behind gate nine back there.
Person:	The gate nine.
Clerk:	Behind gate nine.

These examples thus show that the typical separation and sequential treatment of the processes of "understanding" and "generation" is rather artificial. Chapter 4 showed how a system prompt and user clarification could be handled similarly within the meta-plan framework.

Finally, the ideas developed in this dissertation suggest directions for research outside the area of natural language. The problem solving concerns of this work are very different than those typically addressed by blocks world planners. For example, non-linguistic planners typically ignore issues of multiple goals, whether concurrent (as in the works of Wilensky [97] and Appelt [8]) or hierarchical (as in this work). Related to this difference in emphasis is a need for new work in plan representation, for example development of a fully adequate vocabulary to support concepts such as meta-planning. Chapter 6 also pointed out work in the area of knowledge representation that was necessary to support systems emphasizing the process of plan recognition.

3. Conclusion

This dissertation has developed a computational theory of dialogue understanding that extends and integrates plan-based and linguistic-based approaches to issues of discourse. Such a framework has enabled the handling of difficult discourse level phenomena such as interrupting subdialogues while maintaining the computational advantages of the plan-based approach.

Bibliography

Conference Abbreviations:

AAAI: Proceedings of the National Conference on Artificial Intelligence

ACL: Proceedings of the Annual Meeting of the Association for Computational Linguistics

Coling: Proceedings of the International Conference on Computational Linguistics

IJCAI: Proceedings of the International Joint Conference on Artificial Intelligence

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.
2. L. Aiello and G. Leui, The Uses of Metaknowledge in AI Systems, *Proceedings of the Sixth European Conference on Artificial Intelligence*, Pisa, September 5-7, 1984, 705-717.
3. J. F. Allen and C. R. Perrault, Analyzing Intention in Utterances, *Artificial Intelligence 15*, 3 (1980), 143-178.
4. J. F. Allen, A. M. Frisch and D. J. Litman, ARGOT: The Rochester Dialogue System, *AAAI*, Pittsburgh, 1982, 67-70.
5. J. F. Allen and J. A. Koomen, Planning Using a Temporal World Model, *IJCAI*, Karlsruhe, 1983, 741-747.
6. J. F. Allen, M. Giuliano and A. M. Frisch, The HORNE Reasoning System, Tech. Rep. 126 revised, University of Rochester, September 1984.
7. J. F. Allen, Towards a General Theory of Action and Time, *Artificial Intelligence 23*, 2 (1984), 123-154.
8. D. E. Appelt, *Planning Natural Language Utterances to Satisfy Multiple Goals*, PhD Thesis,Stanford University, 1981.
9. J. L. Austin, *How To Do Things With Words*, Oxford University Press, New York, 1962.
10. G. E. Barton, A Multiple-Context Equality-based Reasoning System, Tech. Rep. 715, MIT, June 1983.
11. R. J. Brachman, R. J. Bobrow, P. R. Cohen, J. W. Klovstad, B. V. Webber and W. A. Woods, Research in Natural Language Understanding: Annual Report, Report 4274, BBN, 1979.
12. J. Brown and R. Burton, Semantic Grammar: A Technique for Constructing Natural Language Interfaces to Instructional Systems, Report 3587, BBN, May 1977.
13. B. Bruce, Analysis of Interacting Plans as a Guide to the Understanding of Story Structure, *Poetics 9*, (1980), 295-311.

14. B. Bruce, Belief Systems and Language Understanding, in *Trends in Linguistics, Studies and Monographs 19*, Walter de Gruyter and Company, New York, 1983, 113-160.

15. S. Carberry, Tracking User Goals in an Information-Seeking Environment, *AAAI*, Washington, D.C., August 1983, 59-63.

16. M. S. Carberry, Understanding Pragmatically Ill-Formed Input, *Coling84*, Stanford, 1984, 200-206.

17. S. Carberry, A Pragmatics-Based Approach to Understanding Intersentential Ellipsis, *ACL*, Chicago, July 1985, 188-197.

18. J. G. Carbonell, Meta-Language Utterances in Purposive Dialogues, CMU-CS-82-125, CMU, June 19, 1982.

19. J. G. Carbonell and P. J. Hayes, Recovery Strategies for Parsing Extragrammatical Language, *AJCL 9*, 3-4 (July-December 1983), 123-146.

20. N. F. Carver, V. R. Lesser and D. L. McCue, Focusing in Plan Recognition, *AAAI*, Austin, Texas, August 1984, 42-48.

21. E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.

22. P. R. Cohen and C. R. Perrault, Elements of a Plan-Based Theory of Speech Acts, *Cognitive Science 3*, 3 (1979), 177-212.

23. P. R. Cohen, C. R. Perrault and J. F. Allen, Beyond Question Answering, in *Strategies for Natural Language Processing*, W. Lehnert and M. Ringle (ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1982, 245-274.

24. R. Cohen, A Computational Model for the Analysis of Arguments, Ph.D. Thesis and Tech. Rep. 151, University of Toronto, October 1983.

25. R. Cohen, A Computational Theory of the Function of Clue Words in Argument Understanding, *COLING-84*, Stanford, July 1984, 251-258.

26. P. R. Cohen and H. J. Levesque, Speech Acts and Rationality, *ACL*, Chicago, July 1985, 49-60.

27. R. E. Cullingford, Script Application: Computer Understanding of Newspaper Stories, Research Report #116, Yale University, 1978.

28. K. Donnellan, Reference and Definite Descriptions, *Philosophical Review LXXV*, (1966), 281-304.

29. M. G. Dyer, The Role of Affect in Narrative, *Cognitive Science 7*, 3 (1983), 211-242.

30. R. E. Fikes and N. J. Nilsson, STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence 2*, 3/4 (1971), 189-208.

31. M. R. Genesereth, The Role of Plans in Automated Consultation, *IJCAI*, Tokyo, 1979, 311-319.

32. B. A. Goodman, Communication and Miscommunication, PhD Thesis, University of Illinois, Urbana, 1984.

33. B. A. Goodman, Repairing Reference Identification Failures by Relaxation, *ACL*, Chicago, July 1985, 204-217.

34. R. H. Granger, Scruffy Text Understanding: Design and Implementation of 'Tolerant' Understanders, *ACL*, Toronto, June 1982, 157-160.

35. H. P. Grice, Meaning, *Philosophical Review LXVI*, (1957), 377-388.

36. J. E. Grimes, *The Thread of Discourse*, Mouton, The Hague, 1975.

37. B. J. Grosz, The Representation and Use of Focus in Dialogue Understanding, Technical Note 151, SRI, July 1977.

38. B. J. Grosz, Utterance and Objective: Issues in Natural Language Communication, *IJCAI*, Tokyo, 1979, 1067-1076.

39. B. J. Grosz, A. K. Joshi and S. Weinstein, Providing a Unified Account of Definite Noun Phrases in Discourse, *ACL*, MIT, June 1983, 44-50.

40. B. J. Grosz and C. L. Sidner, Discourse Structure and the Proper Treatment of Interruptions, *IJCAI*, Los Angeles, August 1985, 832-839.

41. R. Guindon, Anaphora Resolution: Short-Term Memory and Focusing, *ACL*, Chicago, July 1985, 218-227.

42. M. A. K. Halliday, Language Structure and Language Function, in *New Horizons in Linguistics*, J. Lyons (ed.), Penguin, New York, 1970.

43. M. A. K. Halliday, *System and Function in Language*, Oxford University Press, London, 1976.

44. G. G. Hendrix, Human Engineering for Applied Natural Language Processing, *IJCAI-77*, MIT, August 1977, 183-191.

45. G. G. Hendrix, Encoding Knowledge in Partitioned Networks, in *Associative Networks*, N. V. Findler (ed.), Academic Press, New York, 1979, 51-92.

46. J. R. Hobbs, Coherence and Coreference, *Cognitive Science 3*, 1 (1979), 67-90.

47. M. K. Horrigan, Modelling Simple Dialogs, Master's Thesis, Technical Report Number 108, University of Toronto, May 1977.

48. G. Jefferson and J. Schenkein, Some Sequential Negotiations in Conversation, in *Studies in the Organization of Conversational Interaction*, J. Schenkein (ed.), Academic Press, New York, 1978.

49. G. Jefferson, Sequential Aspects of Storytelling in Conversation, in *Studies in the Organization of Conversational Interaction*, J. Schenkein (ed.), Academic Press, New York, 1978.

50. M. W. Kahrs, A. R. Haas and D. M. Russell, Transcripts of Task-Oriented Cooking Dialogues, Unpublished Manuscript, University of Rochester, 1979.

51. H. Kautz, Towards a Theory of Plan Recognition, Tech. Rep. 162, University of Rochester, July 1985.

52. W. Kintsch and T. A. VanDijk, Toward a Model of Text Comprehension and Production, *Psychological Review 85*, (1978), 363-394 .

53. S. Kuno, Generative Discourse Analysis in America, in *Current Trends in Textlinguistics*, W. U. Dressler (ed.), Walter de Gruyter, Berlin, 1977 .

54. D. Levy, Communicative Goals and Strategies: Between Discourse and Syntax, in *Syntax and Semantics*, vol. 12, T. Givon (ed.), Academic Press, New York, 1979, 183-212.

55. C. Linde, The Organization of Discourse, in *The English Language in its Social and Historical Context*, Shopen, Zwicky and Griffen (ed.), .

56. D. J. Litman, Discourse and Problem Problem Solving, Report 5338, Bolt Beranek and Newman, July 1983.

57. D. J. Litman and J. F. Allen, A Plan Recognition Model for Clarification Subdialogues, *Coling84*, Stanford, July 1984, 302-311.

58. D. J. Litman and J. F. Allen, A Plan Recognition Model for Subdialogues in Conversation, Tech. Rep. 141, University of Rochester, November 1984.

59. R. E. Longacre, *An Anatomy of Speech Notions*, The Peter de Ridder Press, Lisse, 1976.

60. W. C. Mann, Discourse Structures for Text Generation, *Coling84*, Stanford, July 1984, 367-375.

61. D. A. McAllester, Reasoning Utility Package User's Manual, AI Memo 667, MIT, 1982.

62. D. A. McAllester, Solving Uninterpreted Equations with Context-Free Expression Grammars, AI Memo 708, MIT, 1983.

63. J. McCarthy and P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in *Machine Intelligence 4*, B. Meltzer and D. Michie (ed.), Edinburgh University Press, Edinburgh, 1969, 463-502.

64. K. R. McKeown, *Generating Natural Language Text in Response to Questions about Database Structure*, PhD Thesis, University of Pennsylvania, Philadelphia, 1982.

65. K. R. McKeown, M. Wish and K. Matthews, Tailoring Explanations for the User, *IJCAI*, Los Angeles, August 1985, 794-798.

66. M. Minsky, A Framework for Representing Knowledge, in *Psychology of Computer Vision*, P. H. Winston (ed.), McGraw-Hill, New York, 1975.

67. A. Newell and H. A. Simon, GPS, A Program that Simulates Human Thought, in *Computers and Thought*, E. Feigenbaum and J. Feldman (ed.), McGraw-Hill, New York, 1963, 279-293.

68. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.

69. Z. Palkova and B. Palek, Functional Sentence Perspective and Textlinguistics, in *Current Trends in Textlinguistics*, W. U. Dressler (ed.), Walter de Gruyter, Berlin, 1977 .

70. L. Polanyi and R. J. H. Scha, On the Recursive Structure of Discourse, *Proceedings of the Tilsburg Symposium on Connectedness in Sentences, Text, and Discourse*, Tilsburg, January 1982.

71. L. Polanyi and R. J. H. Scha, The Syntax of Discourse, *Text (Special Issue: Formal Methods of Discourse Analysis) 3*, 3 (1983), 261-270.

72. L. Polanyi and R. Scha, A Discourse Model for Natural Language, in *The Structure of Discourse*, Ablex, Forthcoming.

73. M. Pollack, Inferring Domain Plans in Question Answering, PhD Dissertation, University of Pennsylvania, Forthcoming.

74. H. Pople, On the Mechanization of Abductive Logic, *IJCAI*, Stanford, August 1973, 147-152.

75. R. Reichman, Conversational Coherency, *Cognitive Science 2*, 4 (1978), 283-328.

76. R. Reichman, Plain Speaking: A Theory and Grammar of Spontaneous Discourse, Report No. 4681, Bolt, Beranek and Newman, 1981.

77. R. Reichman-Adar, Extended Person-Machine Interfaces, *Artificial Intelligence 22*, 2 (1984), 157-218.

78. R. Reiter, A Logic for Default Reasoning, *Artificial Intelligence 13*, (1980), 81-132.

79. E. D. Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence 5*, 2 (1974), 115-135.

80. E. D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier, New York, 1977.

81. H. Sacks, E. A. Schegloff and G. Jefferson, A Simplest Systematics for the Organization of Turn-Taking for Conversation, *Language 50*, 4, Part 1 (December 1974), 696-735.

82. R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals, and Understanding*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

83. C. F. Schmidt, N. S. Sridharan and J. L. Goodson, The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence, *Artificial Intelligence 11*, (1978), 45-83.

84. J. R. Searle, in *Speech Acts, an Essay in the Philosophy of Language*, Cambridge University Press, New York, 1969.

85. J. R. Searle, Indirect Speech Acts, in *Speech Acts*, vol. 3, P. Cole and Morgan (ed.), Academic Press, New York, NY, 1975.

86. C. L. Sidner and D. J. Israel, Recognizing Intended Meaning and Speakers' Plans, *IJCAI*, Vancouver, 1981, 203-208.

87. C. L. Sidner, Protocols of Users Manipulating Visually Presented Information with Natural Language, Report 5128, Bolt Beranek and Newman , September 1982.

88. C. L. Sidner, Focusing in the Comprehension of Definite Anaphora, in *Computational Models of Discourse*, M. Brady (ed.), MIT Press, Cambridge, 1983, 267-330.

89. C. L. Sidner and M. Bates, Requirements of Natural Language Understanding in a System with Graphic Displays, Report Number 5242, Bolt Beranek and Newman Inc., March 1983.

90. C. L. Sidner, Plan Parsing for Intended Response Recognition in Discourse, *Computational Intelligence 1*, 1 (February 1985), 1-10.

91. M. Stefik, Planning with Constraints (MOLGEN: Part 1), *Artificial Intelligence 16*, (1981), 111-140.

92. A. Tate, Generating Project Networks, *IJCAI-5*, MIT, 1977, 888-893.

93. S. A. Thompson, The Passive in English: A Discourse Perspective, Linguistics Dept. Working Paper, UCLA, Summer 1983.

94. D. L. Waltz and B. A. Goodman, Writing a Natural Language Data Base System, *IJCAI-77*, MIT, August 1977, 144-150.

95. R. M. Weischedel and N. K. Sondheimer, An Improved Heuristic for Ellipsis Processing, *ACL*, Toronto, June 1982, 85-88.

96. R. Wilensky, Understanding Goal-Based Stories, Research Report #140, PhD Thesis, Yale University, September 1978.

97. R. Wilensky, *Planning and Understanding*, Addison-Wesley Publishing company, Reading, Massachusetts, 1983.

98. T. Winograd, *Understanding Natural Language*, Academic Press, NY, 1972.

99. B. Woolf and D. D. McDonald, Context-Dependent Transitions in Tutoring Discourse, *AAAI*, Austin, Texas, August 1984, 355-361.

END

DTIC

9-86